CERTIFIED COPY OF PRIORITY DOCUMENT

Patent Office
Canberra

I, ANNA MAIJA MADL, ACTING TEAM LEADER EXAMINATION SUPPORT & SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PP 9961 for a patent by SILVERBROOK RESEARCH PTY LTD filed on 23 April 1999.

WITNESS my hand this
Twenty-first day of October 1999

A.M. Madl.

ANNA MAIJA MADL
ACTING TEAM LEADER
EXAMINATION SUPPORT & SALES

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION

Applicant(s):

SILVERBROOK RESEARCH PTY LTD

Invention Title:

AN IMAGE CREATION METHOD AND APPARATUS (CEP04)

The invention is described in the following statement:

AN IMAGE CREATION METHOD AND APPARATUS (CEP04)

Field of the Invention

The present invention relates to the field of printing and, in particular, discloses a controller for a double sided ink-jet printing system having page width print heads.

Background of the Invention

Many different types of printers are generally known. For example, offset printers, ink-jet prints, laser printers, etc., are utilised to provide for output printing on output media.

With any printer device, it is desirable to provide for as inexpensive a form of printing as possible. Also, it is often desirable to have an extremely compact arrangement in addition to printing on both sides of an output print media such as paper sheets. Ideally, a printer controller for controlling the overall system is also provided.

Summary of the Invention

It is an object of the present invention to provide an effective printer controller for controlling an improved compact double sided ink jet printing system.

In accordance with a first aspect of the present invention, there is provided a double sided printing controller for printing on both sides of a print media comprising: a first printer controller unit for controlling a first pagewidth ink jet print head for printing a first surface on the print media; a second printer controller unit for controlling a second pagewidth ink jet print head for simultaneously printing on a second surface of the print media; an inter-communications unit interconnected between the first controller unit and the second controller unit for synchronizing the first and second printer controller units; and host interface for interconnecting with a host computer system for receipt of images to be printed by the printer controller units.

Pages to be printed by the second printer

controller are preferably forwarded initially to the first printer controller unit and then from the first controller unit to the second controller unit for printing by the second controller unit.

The first and second printer controller unit can be each preferably each formed from substantially identical circuitry.

Description Preferred in Other Embodiments

The preferred embodiment of the present invention is as set out in the attached appendix B which provides for a detailed description of the implementation of a printer system denoted CEPRINT, including the adaptation to a two side printer system utilising a page width ink-jet print head.

The ink jet technology utilized can be a wide pagewidth printing system as disclosed in the attached appendix A.

It would be appreciated by a person skilled in the art that numerous variations and/or modifications may be made to the present invention as shown in the specific embodiments in the attached documentation without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects to be illustrative and not restrictive.

The following numbered paragraphs set out various aspects of the present invention:

1. A double sided printing controller for printing on both sides of a print media comprising:

a first printer controller unit for controlling a first pagewidth ink jet print head for printing a first surface on said print media;

a second printer controller unit for controlling a second pagewidth ink jet print head for simultaneously printing on a second surface of said print media;

an inter-communications unit interconnected between said first controller unit and said second controller unit for synchronizing said first and second printer controller units; and

host interface for interconnecting with a host computer system for receipt of images to be printed by said printer controller units.

2. A double sided printing controller as set out in paragraph 1 wherein pages to be printed by said second printer controller are forwarded initially to said first printer controller unit and then from said first controller unit to said second controller unit for printing by said second controller unit.

3. A double sided printing controller as set out in any previous paragraph wherein said first and second printer controller unit include are each formed from substantially identical circuitry.

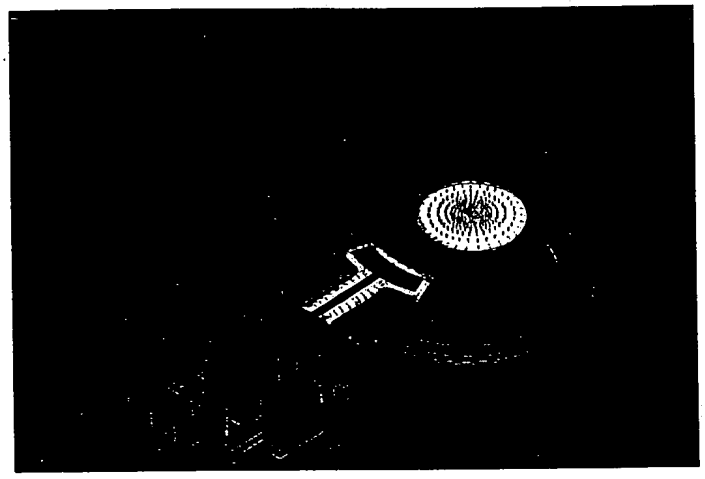
4. A print controller for controlling double sided printing substantially as hereinbefore described with reference to the accompanying appendicies.

ABSTRACT

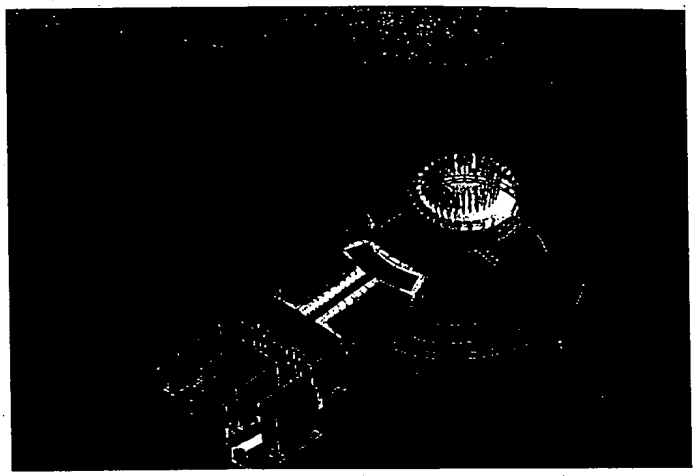
A double sided printing controller for printing on both sides of a print media comprising: a first printer controller unit for controlling a first pagewidth ink jet print head for printing a first surface on the print media; a second printer controller unit for controlling a second pagewidth ink jet print head for simultaneously printing on a second surface of the print media; an inter-communications unit interconnected between the first controller unit and the second controller unit for synchronizing the first and second printer controller units; and host interface for interconnecting with a host computer system for receipt of images to be printed by the printer controller units. Pages to be printed by the second printer controller are preferably forwarded initially to the first printer controller unit and then from the first controller unit to the second controller unit for printing by the second controller unit.

MEMJET

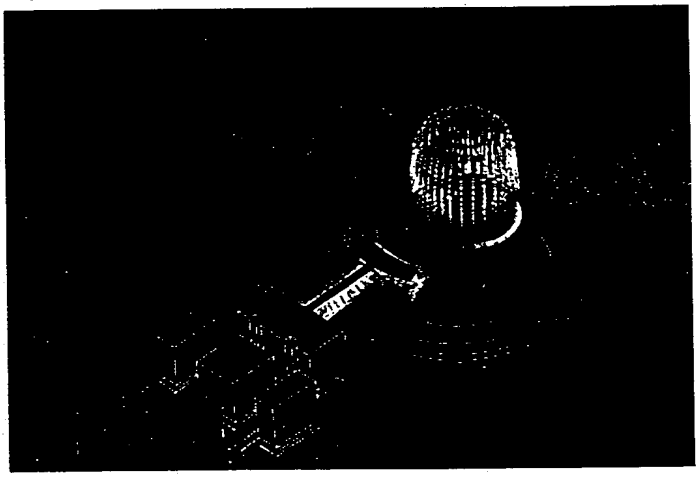
Micro Electro Mechanical Inkjet



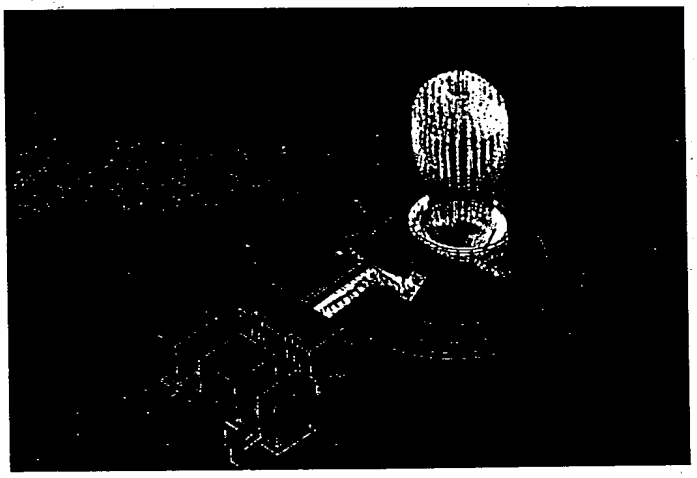
0 μs



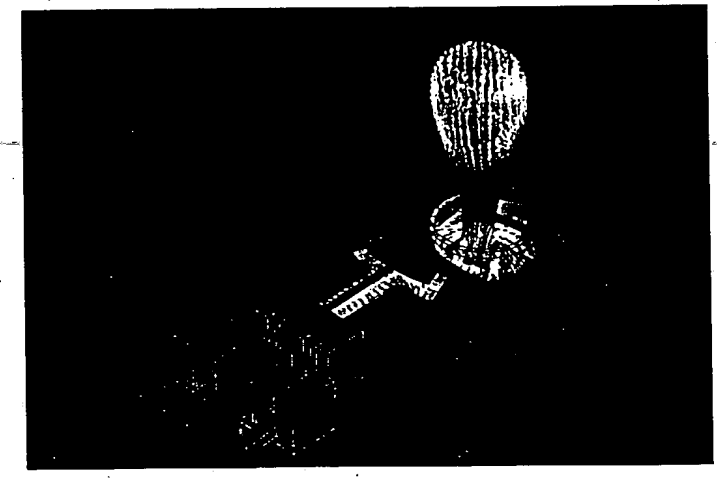
1 μs



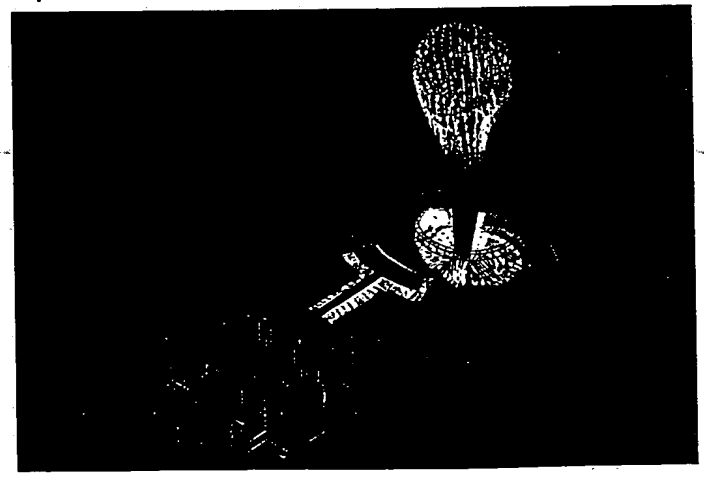
2 μs



3 μs



4 μs



5 μs

Silverbrook Research Pty Ltd
393 Darling Street, Balmain
NSW 2041 Australia
Phone: +61 2 9818 6633
Fax: +61 2 9818 6711

MEMJET INTRODUCTION

Memjet is a new digital printing technology under development at Silverbrook Research. It is a 'clean slate' development aimed at developing the 'perfect' print technology for a wide range of applications for which current digital printing technologies are inadequate. The price/performance advantage over existing technologies such as laser printers and thermal inkjet printers is around two orders of magnitude.

Key Features of Memjet:

- Pagewidth inkjet printing - no scanning printheads, therefore very fast.
- High nozzle count: 51,200 nozzles for A4/letter.
- Full quality color photographic images at 1600 dpi.
- Full quality text, including Japanese Kanji.
- Wide ink and paper flexibility - three times less water than 600 dpi printers.
- High speed - 30 pages per minute (ppm) to 4000 ppm.
- High nozzle density - 25,600 nozzles in a 75mm^2 chip (compare to latest HP - 300 nozzles in 75 mm^2).
- Low cost - under $10 for a 30 ppm letter/A4 full color 1600 dpi printhead.
- Low power allows battery operation for many applications.
- Small size - printers incorporated in mobile phones, cameras, even pens.
- Simple drive circuits - 3V digital ASIC with low pincount.
- High volume manufacture - 56 million 8" heads from a 25,000 wafer per month fab.
- Low manufacturing investment -can adapt a 0.5 micron CMOS fab.
- Excellent patent protection - 220 US patents pending, both basic patents and strategic blocking patents.

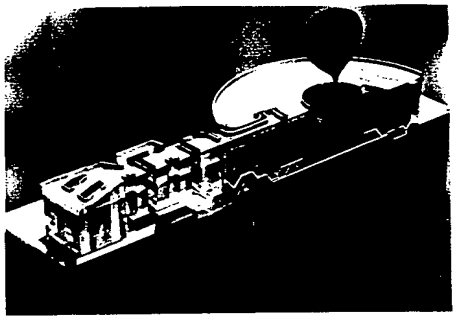
Radical, not Evolutionary

The closest technology to Memjet is Thermal Ink Jet (TIJ). This technology was invented roughly simultaneously by Canon and Hewlett-Packard around 1980 (Canon's variety is know as 'Bubblejet'), and currently results in annual revenues around $24 billion.

Thermal inkjet printheads propel a droplet of ink out of a nozzle by superheating a tiny volume of ink. This ink undergoes a flash evaporation process, forming a bubble which pushes the ink out of the nozzle.

There has been a massive investment in developing thermal inkjet technology, which has steadily advanced products from initial 200 dpi black only printers with 12 nozzles, to current full color 600 dpi printers containing printheads with as many as 608 nozzles.

However, Memjet does not use the thermal inkjet operating principle, and is not an extension of TIJ technology.



Cross section of a Memjet nozzle

Micro Electro Mechanical Systems (MEMS)

Memjet is derived from MEMS technology. MEMS is Micro Electro Mechanical Systems, and is basically the construction of mechanical systems using VLSI chip fabrication techniques. MEMS allows the integration of hundreds of millions of mechanical devices on a wafer. Certain MEMS processes (such as the Memjet process) allow the integration of MEMS and CMOS processing. This is essential for a pagewidth printhead, otherwise around 50,000 off-chip connections would be required, making the cost prohibitive for volume markets.

MEMS devices are used in many applications, though few of these applications have reached substantial volume sales. Some of the best known are

CONTENTS

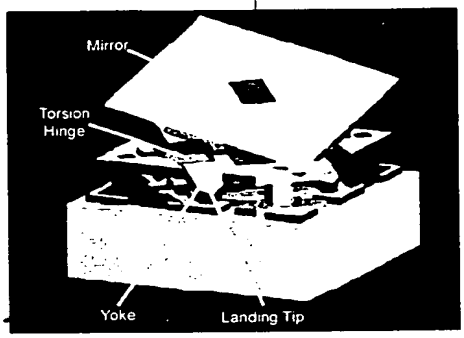
Memjet Introduction	**3**
Advantages	**6**
Yield	**13**
Volume	**15**
Prototype Fabrication	**17**

Silverbrook Research
393 Darling Street
Balmain NSW 2041
Australia
Ph: +61 2 9818 6633
Fax: +61 2 9818 6711
info@silverbrook.com.au

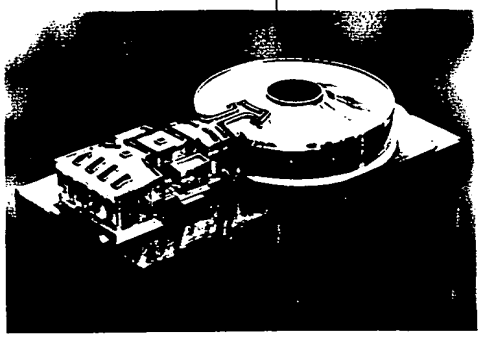
accelerometers for car airbags, and Texas Instruments' DMD (Digital Micromirror Device).

The DMD is an array of around 1 million tiny mirrors on a CMOS chip. Each mirror is independently deflected in response to video image data distributed on the CMOS chip. Each mirror reflects light either towards a viewing screen, or towards a baffle. As all of the mirrors are independently controlled, a large amount of data can be projected even though the individual speed of the mirrors is only around 50 kHz.

Memjet is conceptually similar. A Memjet printhead contains around 25,000 tiny paddles on a CMOS chip. Each paddle is independently deflected in response to page image data distributed on the CMOS chip. When deflected, each paddle pushes a microscopic ink drop out of a nozzle towards the paper. As all of the paddles are independently controlled, a large amount of data can be printed even though the individual speed of the paddles is only around 40 kHz.



Texas Instruments Digital Micromirror Device



Memjet Inkjet nozzle

Memjet Value Proposition

The value proposition of Memjet is 'practical pagewidth inkjet printing'.

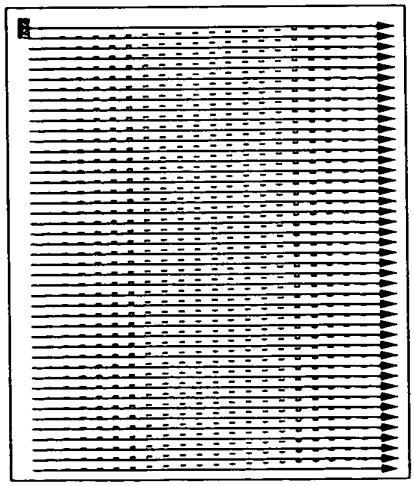
A scanning printhead is fundamentally around 100 times slower than a pagewidth printhead. This is because the a scanning printhead must scan across the page numerous times - around 40 times for draft mode and 200 times for photo quality.

The most fundamental difference between scanning and pagewidth printheads is the width of the printhead.

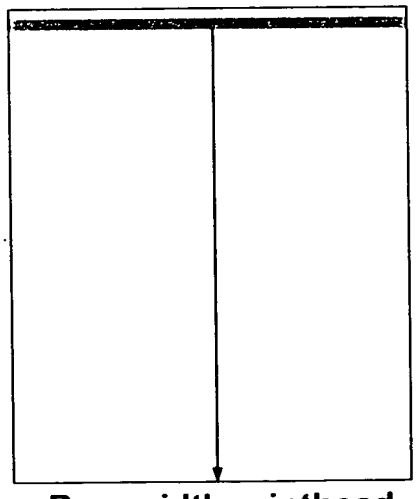
There is an essential divide between scanning printheads (roughly 1/2") and pagewidth printheads (8" for US letter/ A4). Printheads between these values are not very useful - there is no 'evolutionary' progression.

For example: you can't make a letter/ A4 pagewidth printer out of a single 4" printhead. Neither can you make a reasonable scanning printer - a 4" print swath is virtually impossible to 'stitch' on the paper without getting highly objectionable lines across the page.

A pagewidth printhead means lots of nozzles. For a 1600 dpi, 4 color, 8" printhead, 51,200 nozzles are required. This compares to typically 300 nozzles for TIJ, and 128 nozzles for piezoelectric inkjets (such as Epson and Tektronix).



Scanning printhead



Pagewidth printhead

Previous pagewidth printhead projects

Pagewidth printheads have long been the 'holy grail' of the inkjet industry, and have been the subject of intensive research at various large companies for more than a decade.

Previous attempts at pagewidth inkjets have mostly tried to scale up either of the two current technologies - thermal inkjet or piezoelectric inkjet. Both of these approaches fail for a number of fundamental reasons.

Why cant you make pagewidth thermal inkjet printheads?

Thermal inkjet (TIJ or Bubblejet - HP and Canon's technology) is not suitable for pagewidth printheads as the power consumption is around 100 times too high. This makes it very difficult to get sufficient power into the printhead. However, even more difficult is getting the waste heat out of the printhead without boiling the ink.

TIJ cannot be made in a monolithic process, as the nozzle plates must be around 30 microns thick to withstand the transient pressures generated by the bubble. As a result, there are extraordinary manufacturing difficulties in scaling up from small printheads to pagewidth printheads. These include differential thermal expansion (making it extremely difficult to align the nozzles at both ends of a long printhead simultaneously) and cracking of the silicon chips due to the high stresses during nozzle plate attachment.

Also, TIJ and Bubblejet printheads wear out due to kogation and cavitation (neither of these problems affect Memjet or piezoelectric printheads). While it is cost effective to frequently replace 1 cm long scanning printheads, pagewidth printheads should have a longer lifetime.

Why cant you make pagewidth piezoelectric inkjet printheads?

The problems with scaling piezoelectric printheads are entirely different than those for TIJ. The main problem is acoustic crosstalk. As the printhead length increases, the delay between the acoustic pulse of a piezo actuator and its acoustic reflections from the ends of the printhead increase. This causes the pulse to interfere with itself on a timescale which reaches into the critical time of

drop ejection. Also, as the number of nozzles increases, the number of possible interference combinations undergoes a combinatorial explosion. This makes it essentially impossible to obtain consistent drop ejection at various places on the printhead.

Another problem is cost. Silicon is not a piezoelectric material, and it is extremely difficult to integrate piezoelectric materials on CMOS chips. This means that a separate external connection is required for each nozzle. A printhead equivalent to an 8" Memjet printhead would require at least 51,201 external connections. While theoretically possible, the cost of these connections is exorbitant.

The drive voltage required is around 100 V to 200 V, so it is also difficult to integrate large numbers of drivers on a single chip.

Another problem preventing integration is that the piezoelectric material must be electrically poled at around 100,000 Volts, which makes it extremely difficult to protect integrated CMOS circuitry.

How fast can a Memjet printer go?

For most consumer and PC printer applications, the full speed of Memjet is not required. In these applications, the print speed will generally be limited by the low cost paper movement mechanism.

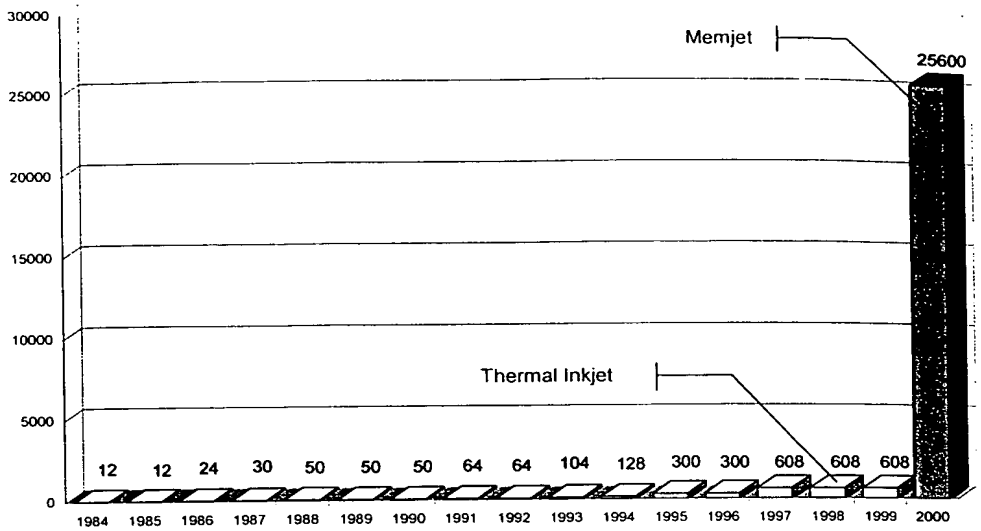
However, for various industrial and commercial printing applications, higher print speeds are desirable.

Individual Memjet nozzles can operate at 40 kHz (and are likely to operate up to 100 kHz). At 1600 dpi, this means a print speed of 25 inches per second. A 'standard' Memjet printhead has four channels for four ink colors. If this printhead is used to print a single color instead, then the maximum print speed is 100 inches per second. A 34 inch wide web (paper roll) fed printer, with printheads on both sides of the paper, printing at 100 inches per second, has an equivalent print speed of 4,364 ppm. At these print speeds, it is possible to compete directly with commercial offset printing.

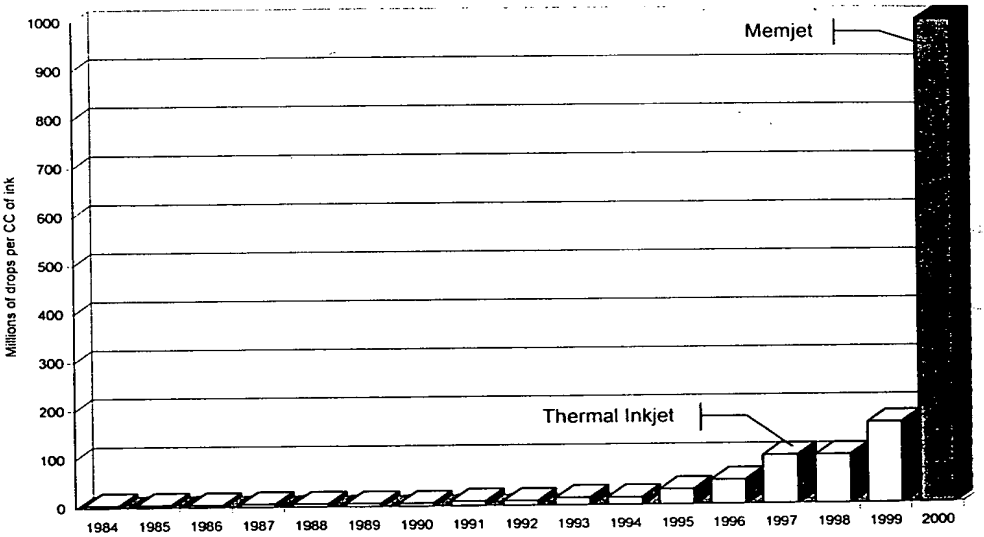
Memjet - TIJ trends

The bar charts to the right show the trends for thermal inkjet printheads from the first commercial products to the present. These trends are compared to what is expected to be the first Memjet printhead - a 4", 4 color, 1600 dpi printhead with 25,600 nozzles. Two such printhead chips butted together make up an 8" pagewidth printhead.

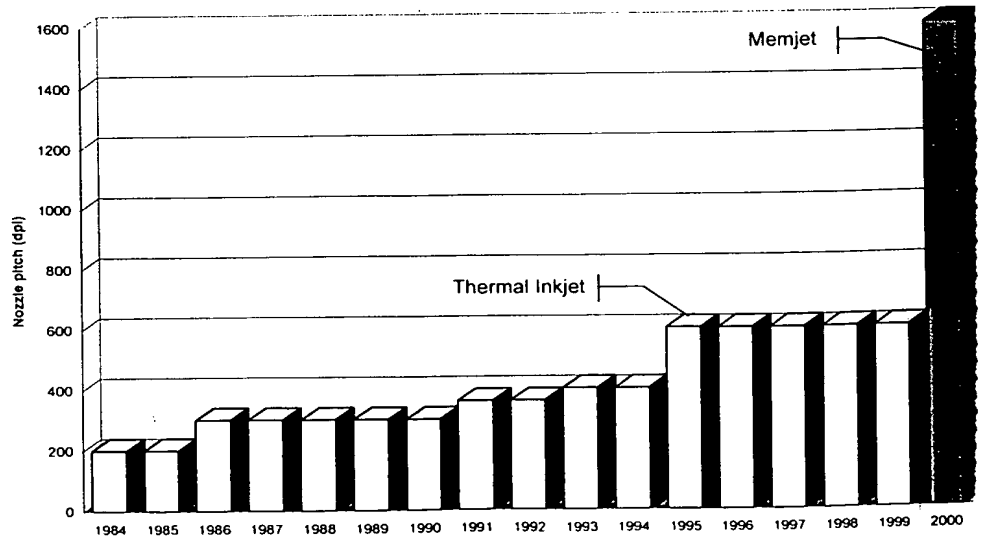
Nozzles per monolithic printhead



Ink drops per CC



Nozzle pitch



ADVANTAGES

The list of advantages is long, and there are no disadvantages listed. This is because Silverbrook Research has spent years analyzing the problems with previous technologies, and ensuring that these problems are solved for Memjet. Memjet has been through 47 major design iterations to converge on its current state, where we believe that further improvements will not be significant. This quest for perfection is ongoing, and it is our intention to correct any undiscovered problems as quickly as possible. As is well known, a problem detected at an early stage can be orders of magnitude cheaper to correct than a problem detected at a prototype or production stage.

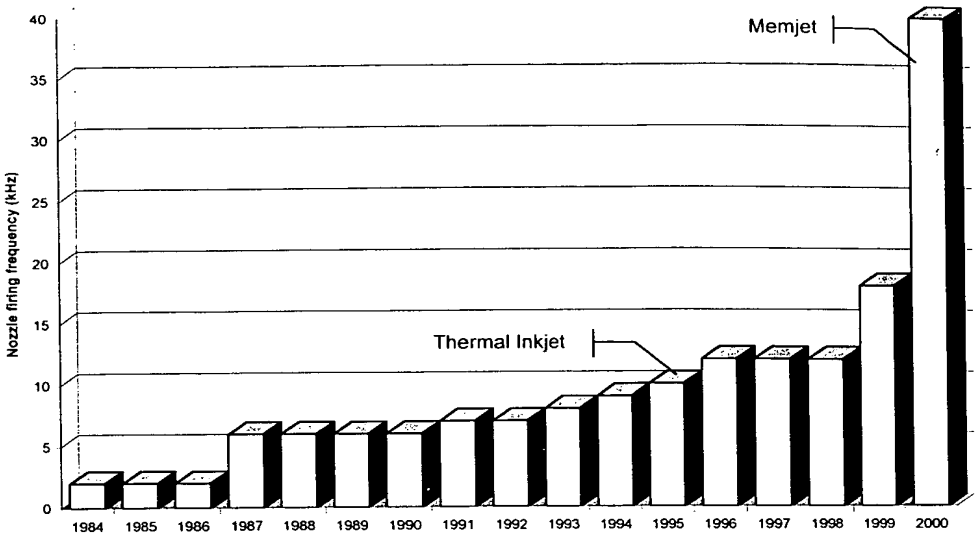
High Resolution

The true resolution of Memjet is 1600 dots per inch (dpi) in both directions. This allows full photographic quality color images, and high quality text (including Kanji). Higher resolutions are possible with the technology. 2400 dpi and 4800 dpi versions have been investigated for special applications, but 1600 dpi is chosen as ideal for most applications. The true resolution of commercially available advanced thermal inkjet devices is around 600 dpi. For piezoelectric systems such as those from Epson the true resolution is substantially lower, but the printhead makes many overlapping passes over each point giving an 'effective resolution' of 1440 dpi, at the expense of speed. 'Addressable resolution' is now often quoted in advertising to give the illusion of higher resolution.

Excellent Image Quality

High image quality requires high resolution and accurate placement of drops. The monolithic pagewidth nature of Memjet allows drop placement to sub-micron precision. High accuracy is also achieved by eliminating misdirected drops, electrostatic deflection, air turbulence, and eddies, and maintaining highly consistent drop volume and velocity. Image quality is also ensured by the provision of sufficient resolution to avoid requiring multiple ink densities. Five color or 6 color 'photo' inkjet systems introduce halftoning artifacts in mid tones (such as fleshtones) if the dye interaction and drop sizes are not absolutely perfect. This problem is eliminated in bi-level three color systems such as Memjet.

Nozzle firing frequency

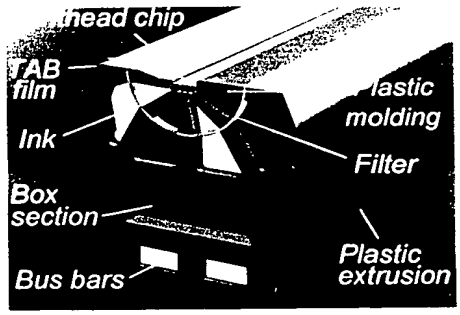


High Speed - up to 120 ppm per printhead

The pagewidth nature of the printhead allows high-speed operation, as no scanning is required. The print speed of a Memjet based printer will usually be determined by market requirements and the paper handling mechanisms required. For most consumer or SOHO applications, 30 ppm printing is around the optimum, as faster printers are dominated by complex and expensive paper handling. To achieve 30 ppm, the Memjet printhead is operated at a drop repetition rate of 10 kHz. The maximum drop repetition rate for Memjet is 40 kHz, allowing 120 ppm full color printing from a single printhead.

Low Cost

A photo-width photographic printhead assembly is projected to cost under $5, and a pagewidth A4 printhead assembly (using two 4" printhead equivalents) is projected to cost less than $10 when manufactured using a 0.5 micron CMOS process on 8" wafers.



When early 300 mm fabs eventually become cost effective, monolithic 8" printheads can be fabricated on a single wafer, reducing production costs further.

All Digital Operation

The high resolution of the printhead is chosen to allow fully digital operation using digital halftoning. This eliminates color non-linearity (a problem with continuous tone printers), and simplifies the design of drive ASICs.

Small Drop Volume

To achieve true 1,600 dpi resolution, a small drop volume is required. Memjet's drop volume is one picoliter (1 pl). The drop size of advanced commercial piezoelectric and thermal inkjet devices is around 10 to 30 pl. This has been steadily reduced over the previous two decades from original drop volumes of around 100 pl. A small drop volume also allows substantially less ink carrier (typically water) to be printed to the page, allowing much faster drying and eliminating print-through. It is theoretically possible to build a 1600 dpi thermal inkjet printhead with 1 pl drops. However, such a printhead would need to print 7.11 times as many drops as a current 600 dpi printer. It would be 7 times slower, and use around 4 times the energy, as a 600 dpi printhead with an equivalent number of nozzles. Thermal inkjet printers are already too slow and use too much power, so it is unlikely that such print-heads will be built. Memjet uses around 100 times less energy to print a drop, and uses around 100 times as many nozzles (51,200 versus around 512 for thermal inkjet), so there are no speed or power problems arising from the high resolution.

Accurate Control of Drop Velocity

As the drop ejector is a precise mechanical mechanism, and does not rely on bubble nucleation, accurate drop velocity control is available. This allows low drop velocities (4 m/s) to be used in applications where media and airflow can be controlled. Drop velocity can be accurately varied over a considerable range by varying the energy provided to the actuator. High drop velocities (10 to 15 m/s) suitable for printing on rough or fibrous surfaces (such as 'plain paper') can be achieved using variations of the nozzle chamber and actuator dimensions. Twelve meters per second is chosen as the nominal velocity for Memjet plain paper applications where airflow is uncontrolled.

Fast Drying

A combination of very high resolution, very small drops, and high dye density allows full color printing with much less water ejected. Memjet ejects around one third of the water of a 600 dpi thermal inkjet printer. This allows fast drying and virtually eliminates paper cockle.

Wide Temperature Range

Memjet is designed to cancel the effect of ambient temperature. Only the change in ink characteristics with temperature affects operation and this can be electronically compensated. Operating temperature range is expected to be 0 °C to 50 °C for water based inks.

No Special Manufacturing Equipment Required

The manufacturing process for Memjet leverages entirely from the established semiconductor manufacturing industry. Most inkjet systems encounter major difficulty and expense in moving from the laboratory to production, as high accuracy specialized manufacturing equipment is required. For Memjet, the equipment required has already been developed at a cost of many billions of dollars for the semiconductor industry.

High Production Capacity Available

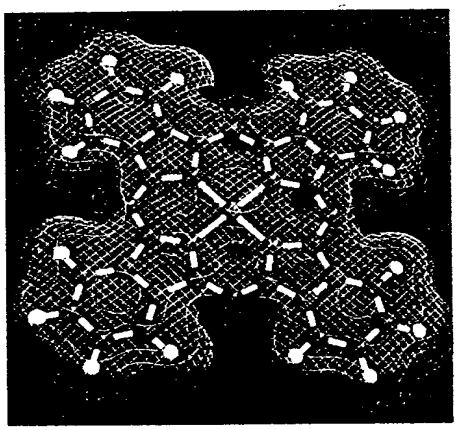
An 8" CMOS fab with 25,000 wafer starts per month can produce around 56 million 8" printheads per annum. There are currently many such CMOS fabs in the world.

Low Factory Setup Cost

The factory set-up cost is low because existing 0.5 micron 6" and 8" CMOS fabs can be used. At the time of projected introduction of the printheads these fabs should be fully amortized, and essentially obsolete for CMOS logic production. Therefore, volume production can use 'old' existing facilities. Most of the MEMS post-processing can also be performed in a CMOS fab, but at least one new piece of equipment will be required, - a deep silicon etcher using the 'Bosch process'. These machines are available from Alcatel, PlasmaTherm, and Surface Technology Systems.

Good Light- Fastness

As the ink is not heated, there are few restrictions on the types of dyes that can be used. This allows dyes to be chosen for optimum light-fastness. Some recently developed dyes from companies such as Zeneca Specialties and BASF have light-fastness of 7 on the blue wool scale. This is equal to the light-fastness of many pigments, and considerably in excess of photographic dyes and of early inkjet dyes.



Copper phthalocyanine - a blue chromophore - showing an electron density isosurface mapped with electric potential

Good Water- Fastness

As with light-fastness, the lack of thermal restrictions on the dye allows selection of dyes for characteristics such as water-fastness. For extremely high water-fastness (as is required for washable fabrics) reactive dyes can be used.

Excellent Color Gamut

The use of transparent dyes of high color purity allows a color gamut considerably wider than that of offset printing and silver halide photography. Offset printing in particular has a restricted gamut due to light scattering from the pigments used. With three-color systems (CMY) or four-color systems (CMYK) the gamut is necessarily limited to the tetrahedral volume between the color vertices. Therefore it is important that the cyan, magenta and yellow dyes are as spectrally pure as possible. A slightly wider 'hexcone' gamut that includes pure reds, greens, and blues can be achieved using a 6 color (CMYRGB) model. Such a six-color printhead can be made economically with a width of only 1 mm.

Elimination of Color Bleed

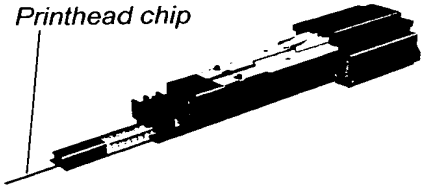
Ink bleed between colors occurs if the different primary colors are printed while the previous color is wet. While image blurring due to ink bleed is typically insignificant at 1600 dpi, ink bleed can 'muddy' the midtones of an image. Ink bleed can be eliminated by using microemulsion-based ink, for which Memjet is highly suited. The use of microemulsion ink can also help prevent nozzle clogging and ensure long-term ink stability.

Advanced Ink Formulations

Silverbrook Research has world-class expertize in quantum chemistry (refer to papers in J. Phys. Chem. and Chemical Physics Letters), and is applying this to the development of a new class of inks with advanced properties.

High Nozzle Count

Memjet has 19,200 nozzles in a monolithic CMY three-color photographic printhead. While this is large compared to other printheads, it is a small number compared to the number of devices routinely integrated on CMOS VLSI chips in high volume production. It is also less than 3% of the number of movable mirrors which Texas Instruments integrates in its Digital Micromirror Device (DMD), manufactured using similar CMOS + MEMS processes.



Cutaway of packaged photographic printhead

51,200 Nozzles per A4 Pagewidth Printhead

A four-color (CMYK) Memjet printhead for pagewidth letter/A4 printing uses two chips. Each 0.65 cm^2 chip has 25,600 nozzles for a total of 51,200 nozzles.

Integration of Drive Circuits

In a printhead with as many as 51,200 nozzles, it is essential to integrate data distribution circuits (shift registers), data timing, and drive transistors with the nozzles. Otherwise, a minimum of 51,201 external connections would be required. This is a severe problem with piezoelectric inkjets, as drive circuits cannot be integrated on piezoelectric substrates. Integration of many millions of connections is common in CMOS VLSI chips, which are fabricated in high volume at high yield. It is the number of off-chip connections that must be limited.

Monolithic Fabrication

Memjet is made as a single monolithic CMOS chip, so no precision assembly is required. All fabrication is performed using standard CMOS VLSI and MEMS processes and materials.

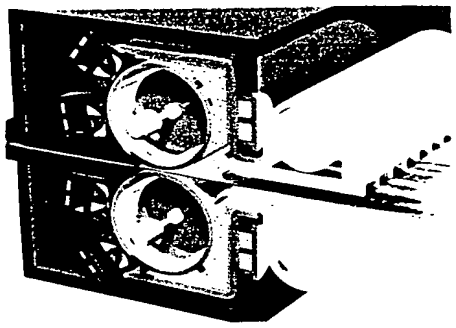
In thermal inkjet and some piezoelectric inkjet systems, the assembly of nozzle plates with the printhead chip is a major cause of low yields, limited resolution, and limited size. Also, pagewidth arrays are typically constructed from multiple smaller chips. The assembly and alignment of these chips is an expensive process.

Modular, Extendable for Wide Print Widths

Long pagewidth printheads can be constructed by butting 'standard' 100 mm Memjet heads together. The edge of the Memjet printhead chip is designed to automatically align to adjacent chips. One printhead gives a photographic size printer, two gives an A4 printer, and four gives an A3 printer. Larger numbers can be used for high speed digital printing, pagewidth wide format printing, and fabric printing.

Low Cost Simultaneous Double Sided Printing

Double sided printing (known as 'duplex' in the office market, and 'perfect' in the commercial print market) can be implemented at low cost simply by including an extra printhead on the other side of the paper, and duplicating the appropriate logic and image processing circuits. The cost and complexity of providing two printheads is less than that of mechanical systems to turn over the sheet of paper.



Simultaneous duplex printhead assembly

Straight Paper Path

As there are no drums required, a straight paper path can be used to reduce the possibility of paper jams. This is especially relevant for office duplex printers, where the complex mechanisms required to turn over the pages are a major source of paper jams.

High Efficiency

Thermal inkjet printheads are only around 0.1% efficient (electrical energy input compared to drop kinetic energy and increased surface energy). Memjet is more than 10 times as efficient.

Self-Cooling Operation

The energy required to eject each drop is 142 nJ (0.142 microJoules), a small fraction of that required for thermal inkjet printers. The low energy allows the printhead to be completely cooled by the ejected ink, with only a 32 °C worst-case ink temperature rise. No heat sinking is required.

Low Pressure

The maximum pressure generated in a Memjet printhead is around 60 kPa (0.6 atmospheres). The pressures generated by bubble nucleation and collapse in thermal inkjet and bubblejet systems are typically in excess of 10 MPa (100 atmospheres), which is 160 times the maximum Memjet pressure. The high pressures in bubblejet and thermal inkjet designs result in high mechanical stresses.

Low Power

A 30 ppm A4 Memjet printhead requires a maximum of 67 Watts when printing full 3-color black. When printing 5% coverage, average power consumption is only 3.4 Watts.

Low Voltage Operation

Memjet can operate from a single 3V supply, the same as typical drive ASICs. Thermal inkjets typically require at least 20 V, and piezoelectric inkjets often require more than 50 V. The Memjet actuator is designed for nominal operation at 2.8 volts, allowing a 0.2 V drop across the drive transistor, to achieve 3V chip operation.

Operation from 2 or 4 AA Batteries

Power consumption is low enough that a photographic Memjet printhead can operate from AA batteries. A typical 6 x 4 inch photograph requires less than 20 Joules to print (including drive transistor losses). Four AA batteries are recommended if the photo is to be printed in 2 seconds. If the print time is increased to 4 seconds, 2 AA batteries can be used.

Battery Voltage Compensation

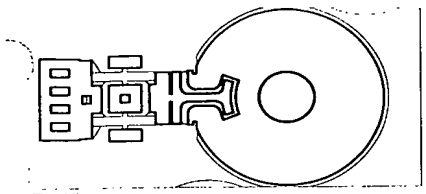
Memjet can operate from an unregulated battery supply, to eliminate efficiency losses of a voltage regulator. This means that consistent performance must be achieved over a considerable range of supply voltages. Memjet senses the supply voltage, and adjusts actuator operation to achieve consistent drop volume.

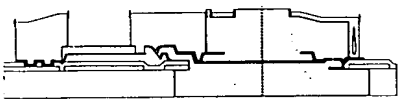
Advanced Nozzle Clearing

Memjet employs two novel patented techniques to clear clogged nozzles, and most conventional techniques can also be used. The degree of potential nozzle clogging is highly dependent on the ink formulation, which depends on the application. In photographic applications clog-proof inks can be used. Hot melt inks can also be completely clog free. When using aqueous inks, it is essential to use nozzle capping and to include a humectant in the ink.

Small Actuator and Nozzle Area

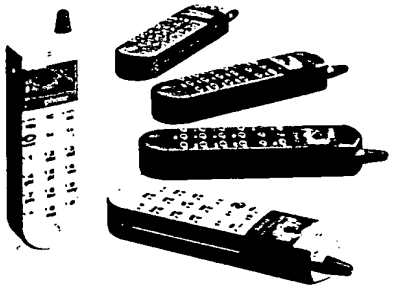
The area required by a Memjet nozzle, actuator, and drive circuit is 3200 μm^2. This is much less than 1% of the area required by piezoelectric inkjet nozzles, and around 1.5% of the area required by TIJ nozzles. The actuator area directly affects the printhead manufacturing cost.





Small Total Printhead Size

An entire printhead assembly (including ink supply channels) for a letter/A4, 30 ppm, 1600 dpi, four color printhead is 210 x 12 x 10 mm. The small size allows incorporation into notebook computers and miniature printers. A photograph printer is 106 x 8 x 8 mm, allowing inclusion in pocket digital cameras, palmtop PCs, mobile phone/fax, and so on. Ink supply channels take most of this volume. The photographic printhead chip itself is only 102 x 0.55 x 0.3 mm.



Printers in 3G mobile phones

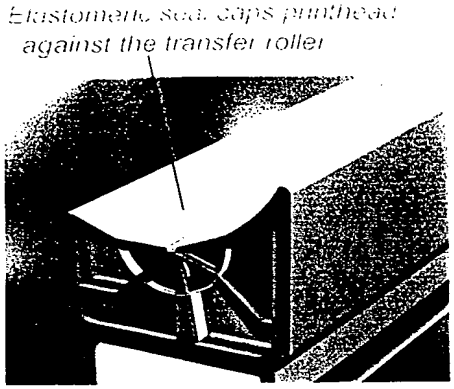
Various Nozzle Capping Systems

Nozzle capping systems have been designed for various applications.

A miniature nozzle capping system has been designed for portable and photographic applications. For a photographic printer this nozzle capping system is only 106 x 5 x 4 mm, and does not require the printhead to move.

For very low cost printers, such as printers incorporated into pens, the nozzle capping mechanism is a single piece of molded plastic costing less than a cent.

In desktop printers, the printhead can be capped against a transfer roller for robust operation and consumer-proofing.



Cross section of packaged printhead (CEprint version)

High Manufacturing Yield

The projected manufacturing yield (at maturity) of the Memjet printheads is at least 80%, as it is primarily a 0.5 micron digital CMOS chip with an area of only 0.15 cm^2 per inch of printhead. Most modern CMOS processes achieve high yield with chip areas in excess of 1 cm^2. For chips less than around 1 cm^2, cost is roughly proportional to chip area. Cost increases rapidly between 1 cm^2 and 4 cm^2, with chips larger than this rarely being practical. There is a strong incentive to ensure that the chip area is less than 1 cm^2.

For thermal inkjet and bubblejet printheads, the chip width is typically around 5 mm, limiting the cost effective chip length to 1 to 2 cm. A major target of Memjet has been to reduce the chip width as much as possible, allowing cost effective monolithic pagewidth printheads.

In the early stages of manufacture, before high yields are obtained, fault tolerance can be used. Although fault tolerance doubles the 'raw' chip area, wafers with defect densities as high as 100 defects per cm^2 can still obtain good printhead yields.

Low Process Complexity

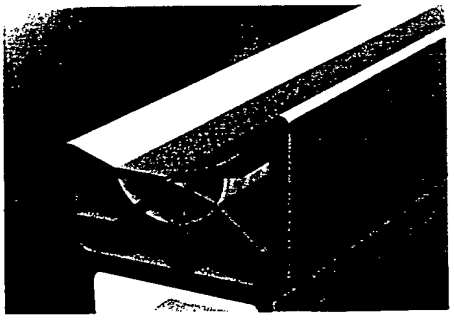
With digital IC manufacture, the mask complexity of the device has little or no effect on the manufacturing cost or difficulty. Cost is proportional to the number of process steps, and the lithographic critical dimensions. Memjet uses a standard 0.5 micron 1P2M CMOS manufacturing process, with an additional 7 MEMS mask steps. This makes the manufacturing process less complex than a typical 0.25 micron CMOS logic process with 5 level metal. However, as the MEMS postprocessing is not standard, a significant amount of process development is required. Considerable effort has been undertaken to minimize the complexity and risk of this process development. However, since any process development is usually difficult and expensive, this is likely to be the highest portion of the remaining development costs.

Simple Testing

Memjet includes test circuits and a test strategy that allows most testing to be completed at the wafer probe stage. Testing of all electrical properties, including the resistance of the actuator, can be completed at this stage. However, actuator motion can only be tested after release from the sacrificial materials. Actuators can be tested before being filled with ink using optical methods. The paddle is reflective and the nozzle chamber is transparent. Paddle deflection can be measured accurately by counting interference fringes of monochromatic light. Final testing of packaged printheads is readily performed by printing a test pattern which is automatically checked using a linear image sensor.

Low Cost Packaging

Memjet is packaged in an injection molded polycarbonate package. All connections are made using Tape Automated Bonding (TAB) technology, though wire bonding can be used as an option. All connections are along one edge of the chip.



Molded plastic package

Relaxed Critical Dimensions

The critical dimension (CD) of the Memjet CMOS drive circuitry is 0.5 microns. Advanced digital ICs such as microprocessors currently use CDs of around 0.25 microns, which is two device generations more advanced than the Memjet printhead requires. Most of the MEMS post processing steps have CDs of 1 micron or greater.

Low Stress during Manufacture

Devices cracking during manufacture are a critical problem with both thermal inkjet and piezoelectric devices. This limits the size of the printhead that it is possible to manufacture. The stresses involved in the manufacture of Memjet printheads are no greater than those required for CMOS fabrication. Memjet printheads are not sawn from the wafer, but are gently plasma etched instead.

No Scan Banding

Memjet is a full pagewidth printhead, so does not scan. This eliminates one of the most significant image quality problems of inkjet printers. Banding due to other causes (mis-directed drops, printhead misalignment) is usually a significant problem in pagewidth printheads. These causes of banding have also been addressed.

'Perfect' Nozzle Alignment

All of the nozzles within a printhead are aligned to sub-micron accuracy by the 0.5 micron stepper used for the lithography of the printhead. Nozzle alignment of two 4" printheads to make an A4 pagewidth printhead is achieved with the aid of mechanical alignment features on the printhead chips. This allows automated mechanical alignment (by simply pushing two printhead chips together) to within 1 micron. If finer alignment is required in specialized applications, 4" printheads can be aligned optically.

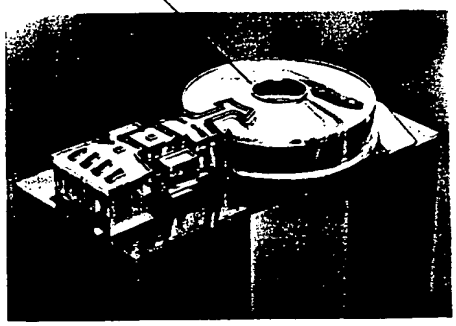
Controllable Drop Velocity

An accurately controlled drop velocity improves image quality, as the position of dots on the moving substrate is more accurate. An accurate drop velocity also enables a lower nominal drop velocity, which reduces power consumption. A low drop velocity requires laminar airflow, with no eddies, to achieve good drop placement on the print medium. This is achieved by the design of the printhead packaging.

For printing on 'rough' surfaces higher drop velocities are desirable. Drop velocities up to 15 m/s can be achieved using variations of the design dimensions. It is possible to manufacture printheads with a 4 m/s drop velocity, and printheads with a 15 m/s drop velocity, on the same wafer. This is because both can be made using the same process parameters.

No Misdirected Drops

Misdirected drops are eliminated by the provision of a thin rim around the nozzle, which prevents the spread of a drop across the printhead surface in regions where the hydrophobic coating is compromised.



No Thermal Crosstalk

When adjacent actuators are energized in bubblejet or other thermal inkjet systems, the heat from one actuator spreads to others, and affects their firing characteristics. In Memjet, heat diffusing from one actuator to adjacent actuators affects both the heater layer and the bend-canceling layer equally, so has no effect on the paddle position. This virtually eliminates thermal crosstalk.

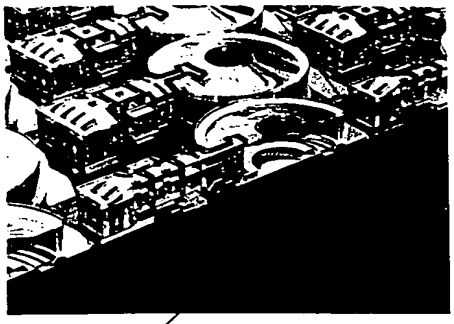
No Structural Acoustic Crosstalk

This is a major problem with piezoelectric printheads. It does not occur in Memjet.

No Fluidic Acoustic Crosstalk

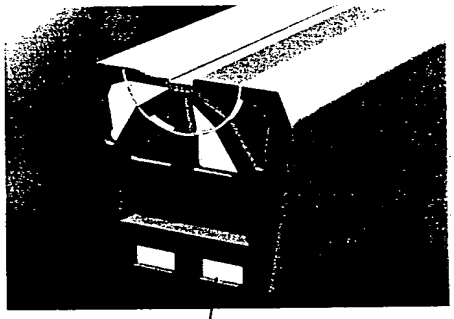
Each simultaneously fired nozzle is at the end of a 300 micron long ink inlet etched through the (thinned) wafer. These ink inlets are connected to large ink channels with low fluidic resistance. This configuration virtually eliminates any effect of drop ejection from one nozzle on other nozzles.

Acoustic crosstalk between 'pods' (groups of nozzles sharing an ink channel) is almost entirely eliminated by the 300 micron long channels opening into relatively large fluid reservoirs. Acoustic crosstalk within a pod is effectively eliminated by the time delay between subsequent drop ejections within a pod. This is typically 20 microseconds, which is more than 40 times longer than the resonant frequency of the ink channel. This gives plenty of time for acoustic vibrations to be damped between drop ejections.



No Power Supply Crosstalk

The thick copper bus-bars in the device package, and the active power supply compensation mechanism, effectively eliminate crosstalk coupled through the power supply.



Permanent Printhead

All of the known problems that limit the life of inkjet printheads have been eliminated, allowing the printheads to be permanently installed. This dramatically lowers the production cost of consumables. Note, however, that the selling price of consumables is not necessarily related to the production cost, and need not be reduced.

No Kogation

Kogation (residues of burnt ink, solvent, and impurities, from the Japanese 'koga' for burnt rice) is a significant problem with bubblejet and other thermal inkjet printheads. Memjet does not have this problem, as the ink is not heated.

No Cavitation

Erosion caused by the violent collapse of bubbles is another problem that limits the life of bubblejet and other thermal inkjet printheads. Memjet does not have this problem because no bubbles are formed.

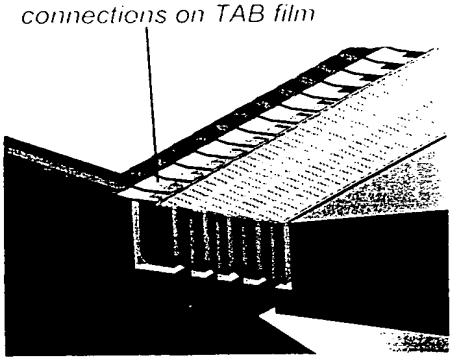
No Electromigration

No metals are used in Memjet actuators or nozzles, which are entirely ceramic. Therefore, there is no problem with electromigration in the actual inkjet devices. The CMOS metallization layers are designed to support the required currents without electromigration. This can be readily achieved because the current considerations arise from heater drive power, not high speed CMOS switching.

Distributed Power Connections

While the energy consumption of Memjet is fifty times less than thermal inkjet, the high print speed and low voltage results in a fairly high electrical current consumption. Worst case current for a photographic Memjet head printing in two seconds from a 3 Volt supply is 4.9 Amps. This is supplied via copper busbars to 256 bond pads along the edge of the chip. Each bond pad carries a maximum of 40 mA. On chip contacts and vias to the drive transistors carry a peak current of 1.5 mA for 1.2 µs, and a maximum average of 12 mA.

Alternating power and signal connections on TAB film



Close-up cross section of packaged printhead

No Corrosion

The nozzle and actuator are entirely formed of glass and titanium nitride (TiN), a conductive ceramic commonly used for metallization barrier layers in CMOS devices. Both materials are at minimum chemical energy levels with respect to water, so do not corrode. Titanium nitride does not corrode or dissolve in extreme environments such as molten aluminum. It is used as the coating for the electrodes in aluminum smelters. TiN is also highly wear resistant - many watches and jewelry items are coated with TiN as it looks like gold, but has much better wear properties.

TiN does slowly oxidize in air above 500 °C, which limits the efficiency of the actuator, as the actuator efficiency is proportional to its temperature rise. Greater efficiency can be obtained by the use of (Ti,Al)N, which has similar properties to TiN but resists oxidation up to 900 °C.

No Electrolysis

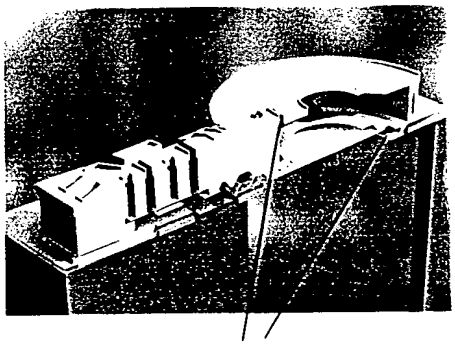
The ink is not in contact with any electrical potentials, so there is no electrolysis. This is achieved by a deliberate design feature in the actuator arm, which is a gap between the actuator loop and the paddle. This electrically isolates the paddle from the drive circuit.

No Fatigue

All actuator movement is within elastic limits, and the materials used are all ceramics, so there is no fatigue. Finite element analysis shows the maximum strain to be 0.5%.

No Friction

No moving surfaces are in contact, so there is no friction.



The actuator and paddle do not contact the nozzle chamber.

No Stiction

'Stiction' is a combination of 'sticking' and 'friction', a problem common to many MEMS devices. Memjet is designed to eliminate stiction during the release of the actuators. This is achieved by the low ratio of width to thickness of the cantilever beam, in combination with the high Young's modulus of the TiN layers.

No Crack Propagation

Finite element analysis (FEA) has shown the maximum strain to be 0.5%. This is well within the crack propagation limit of the actuator, with the typical surface roughness of the TiN layers.

No Electrical Poling Required

Piezoelectric materials must be poled after they are formed into the printhead structure. This poling requires very high electrical field strengths - around 20,000 V/cm. The high voltage requirement typically limits the size of piezoelectric printheads to around 5 cm, requiring 100,000 Volts to pole. Memjet requires no poling.

No Rectified Diffusion

Rectified diffusion - the formation of bubbles due to cyclic pressure variations - is a problem that primarily afflicts piezoelectric inkjets. Memjet is designed to prevent rectified diffusion, as the ink pressure never falls below zero.

Elimination of the Saw Street

The saw street between chips on a wafer is typically 200 microns. This would take 26% of the wafer area. Instead, plasma etching is used, requiring just 4% of the wafer area. This also eliminates breakage during sawing.

Lithography Using Standard Steppers

Although Memjet printhead chips can be as long as the wafer is wide, standard steppers (which typically have an imaging field around 20 mm square) are used. This is because the printhead is 'stitched' using identical half inch exposures. Alignment between stitches fields is not critical, as there are no electrical connections between stitch regions. One segment of each of 32 printheads is imaged with each stepper exposure, giving an 'average' of 4 printheads per exposure.

Integration of Full Color on a Single Chip

Memjet integrates all of the colors required onto a single chip. This cannot be done with pagewidth 'edge shooter' designs, such as Canon Bubblejet.

Wide Variety of Inks

Memjet does not rely on the ink properties for drop ejection. Inks can be based on water, microemulsions, oils, various alcohols, MEK, hot melt waxes, or other solvents. Memjet can be 'tuned' for inks over a wide range of viscosity and surface tension. This is one significant factor allowing a wide range of applications.

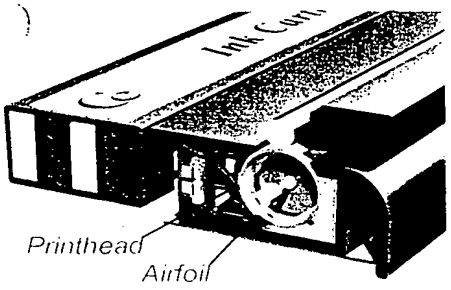
Archival Quality

With the right choice of dye and media, archival permanence significantly better than color photographs can be achieved.

Laminar Air Flow with no Eddies

The printhead packaging is designed to ensure that airflow is laminar, and to eliminate eddies. This is important, as

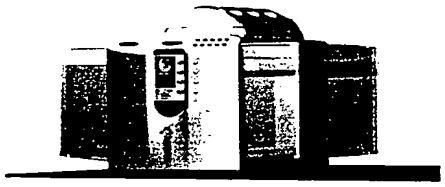
eddies or turbulence could degrade image quality due to the small drop size.



Printer with controlled airflow

High Drop Repetition Rate can be used

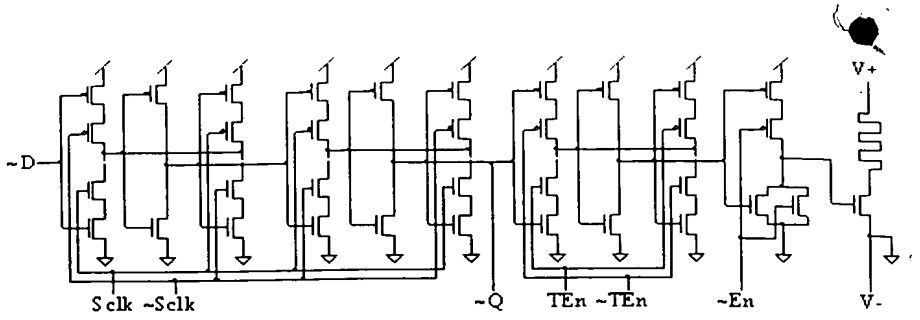
The nominal drop repetition rate of a photographic Memjet is 5 kHz, resulting in a print speed of 2 seconds per photo. The nominal drop repetition rate for an A4 printhead is 10 kHz for 30+ ppm A4 printing. The maximum drop repetition rate is primarily limited by the nozzle refill rate, which is determined by nozzle chamber geometry, flow dynamics, ink pressure, and surface tension. Drop repetition rates of 40 kHz can be achieved (and 100 kHz may be achievable), allowing print speeds of 120 ppm using a single row of nozzles for each color. However, 34 ppm is entirely adequate for most low cost consumer applications.



120 ppm duplex desktop printer

For very high-speed applications, such as commercial printing, multiple printheads can be used in conjunction with fast paper handling. A suitable system for high end commercial printing can use one printhead per color per side. On a 34 inch web, this would give an effective (A4 equivalent) print speed of 10,900 ppm with a 6.25 m/s paper speed.

If the printer speed is reduced, the energy required to print a page is distributed over a longer time, so the power consumption is reduced. For operation from AA batteries this can be important, as the internal resistance of the batteries limits the available power. The battery lifetime is not an important issue, as hundreds of pages can be printed from one set of batteries. The drop repetition rate can be reduced as low as desired to reduce power consumption, as the CMOS design is fully static.



Per-nozzle CMOS circuit

Low Head-to-Paper Speed

The nominal head to paper speed of a photographic Memjet printhead is only 0.076 m/sec. For an A4 printhead it is only 0.16 m/sec., which is about a third of the typical scanning inkjet head speed. The low speed simplifies printer design and improves drop placement accuracy. However, this head-to-paper speed is enough for 34 ppm printing, due to the pagewidth printhead. Higher speeds can readily be obtained where required.

High Speed CMOS not Required

The clock speed of the printhead shift registers is only 14 MHz for an letter/A4 printhead operating at 30 ppm. For a photographic printer, the clock speed is only 3.84 MHz. This is much lower than the speed capability of the CMOS process used. This simplifies the CMOS design, and eliminates power dissipation problems when printing near-white images.

Fully Static CMOS Design

The shift registers and transfer registers are fully static designs. A static design requires 35 transistors per nozzle, compared to around 13 for a dynamic design. However, the static design has several advantages, including higher noise immunity, lower quiescent power consumption, and greater processing tolerances.

Wide Power Transistor

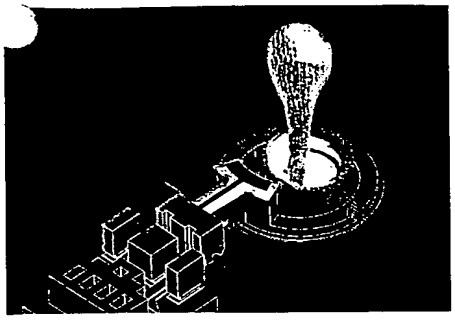
The width to length ratio of the power transistor is 688. This allows a 4 Ohm on-resistance, whereby the drive transistor consumes 6.7% of the actuator power when operating from 3V. This size transistor fits beneath the actuator. Thus an adequate drive transistor, along with the associated data distribution circuits, consumes no chip area that is not already required by the actuator.

There are several ways to reduce the percentage of power consumed by the transistor: increase the drive voltage so that the required current is less, reduce the lithography to less than 0.5 micron, use BiCMOS or other high current drive technology, or increase the chip area, allowing room for drive transistors which are not underneath the actuator. However, the 6.7% consumption of the present design is considered a cost/performance optimum.

Extensive Simulation

Extensive simulations of fluid dynamic, mechanical, thermal, electrical, and other characteristics of the device have been performed. These are done using software developed at Silverbrook Research, in combination with several leading commercial software packages (Ansys, Fidap, and Matlab). Simulation is used as a 'computational microscope' which is able to 'see' microscopic stresses, temperature profiles, and fluid flow in ways impossible with physical experiments. All simulations performed at Silverbrook Research are 'causal'. That is, no assumptions are made about the motion of the ink or other aspects of the device. A simulated voltage pulse is provided to the actuator, and then we watch what happens. Simulations are performed at sufficient resolution to capture the detailed behavior of features such as satellite drops. More than 2,000 simulations have been performed, using more than 20,000 hours

of compute time on several high performance workstations.



Simulated drop ejection of a Memjet printhead

YIELD

Yield is, of course, a critical aspect manufacturing any chips. For this reason, we have put considerable emphasis on maximizing yield. The base CMOS process is 0.5 micron, a mature technology able to achieve very high yields on reasonable sized chips. The chip size is small: about 8 mm^2 per 1/2" print head segment (1.3 cm^2 for an 8" printhead).

The MEMS process itself is not very sensitive to particulate contamination - most feature sizes are well above 1 micron, and most layers are quite insensitive to particulate contamination.

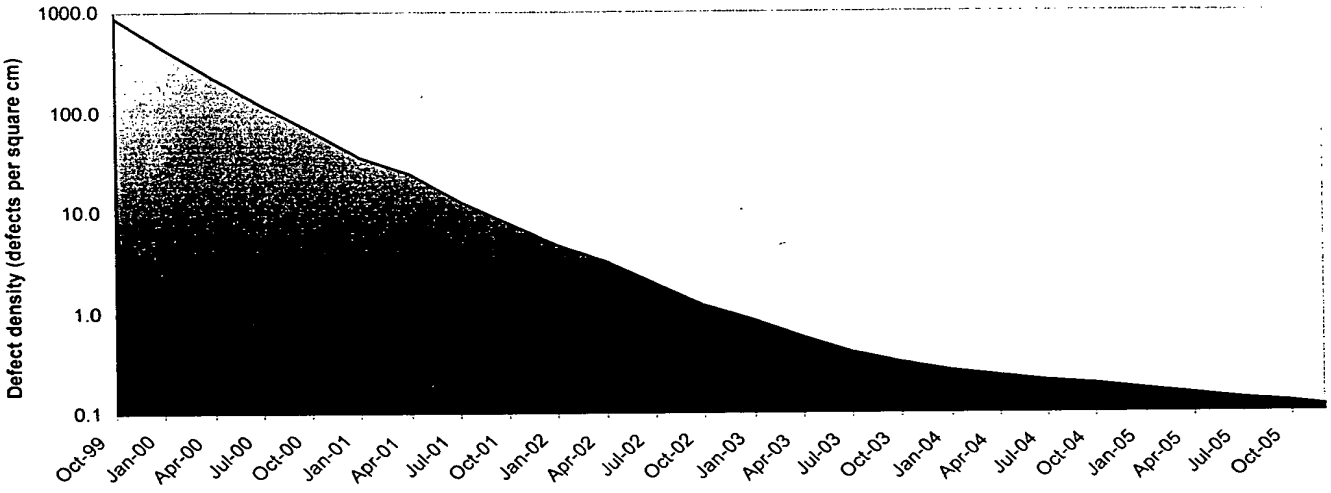
However, it is not a good idea to be complacent about yield, especially for a new process. Accordingly, the yield estimates that we use are pessimistic. The yield is calculated from the defect density for four different printhead configurations.

Yield ramp-up

This projection assumes a slow yield ramp-up over around 5 years. It starts with a defect density of more than 100 defects per cm^2, falling to around 0.2 per cm^2 by 2005.

Defect Density used for these financial projections (log scale)

(note: these defect density projections are intended as a worst case estimate for conservative financial projections - actual defect densities will probably be much lower)



Redundancy and Fault Tolerance

To allow cost-effective production of printheads in the early years when the defect density is high, redundant printheads can be used.

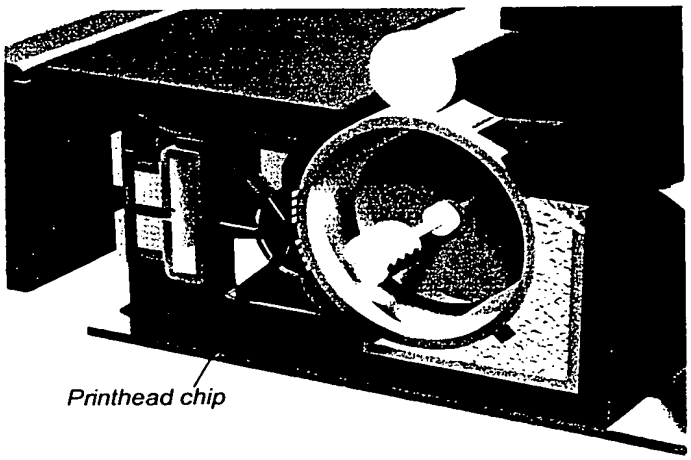
The business model automatically selects between four configurations of print-head, depending upon how the cost of each configuration changes with changing yield. These configurations are:

1) A printhead with full redundancy (two complete rows of nozzles) made from tested and matched 1/2" printhead segments (a total of 32 half inch segments).

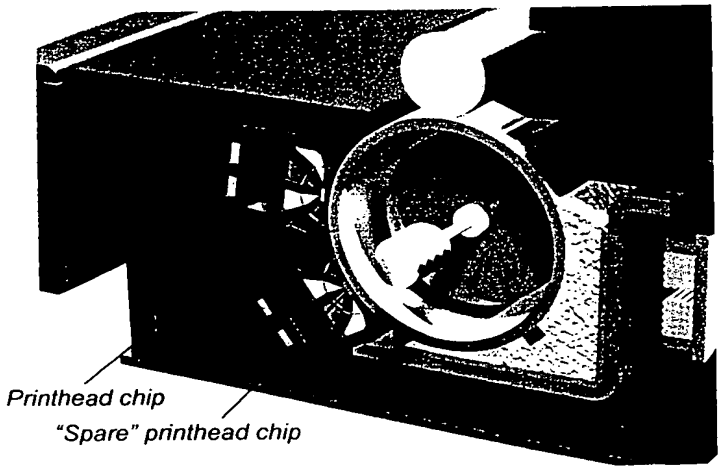
2) A printhead with full redundancy made from two rows of untested 4" printhead segments (a total of 4 four inch segments).

3) A printhead with no redundancy made from 16 half inch tested segments.

4) A printhead with no redundancy made from 2 four inch tested segments.



Normal printhead configuration with a single set of nozzles (51,200 total nozzles)

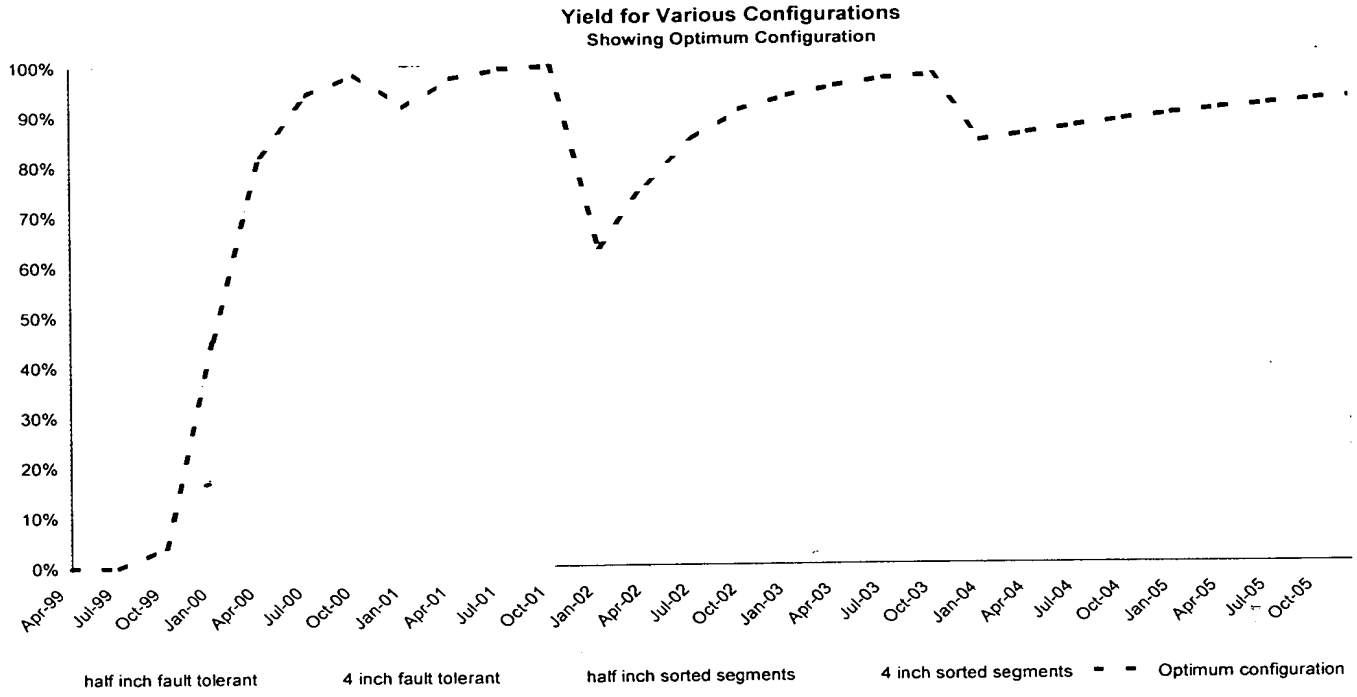


Redundant printhead configuration with two sets of nozzles (102,400 total nozzles)

Printhead Yield

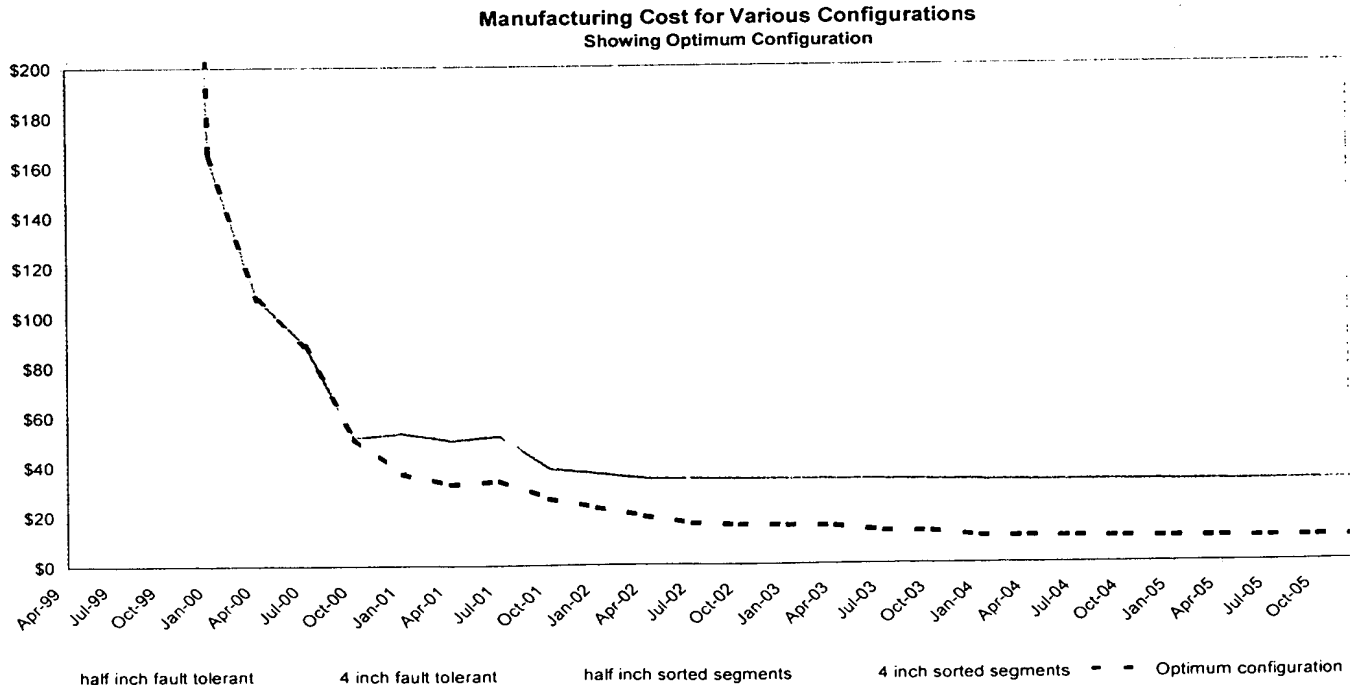
The overall printhead yield can be improved dramatically by the use of redundancy / fault tolerance when the defect densities are high.

The following graph shows the 'sort yield' of 8" printheads of four different types based on the defect density curve above.



Printhead Manufacturing Cost

The manufacturing cost of the printhead changes with yield. Although the yield of the fault tolerant configurations of printhead will always be higher than non-fault tolerant configurations, fault tolerance is not cost effective once the defect densities are low enough. The primary reason for this is that two complete sets of nozzles are required for fault tolerance (unlike memories, where a few redundant rows and/or columns are sufficient). The graph below reflects the total manufacturing costs of printheads, including device sorting and packaging. The volume of manufacture is also considered into the cost projection.



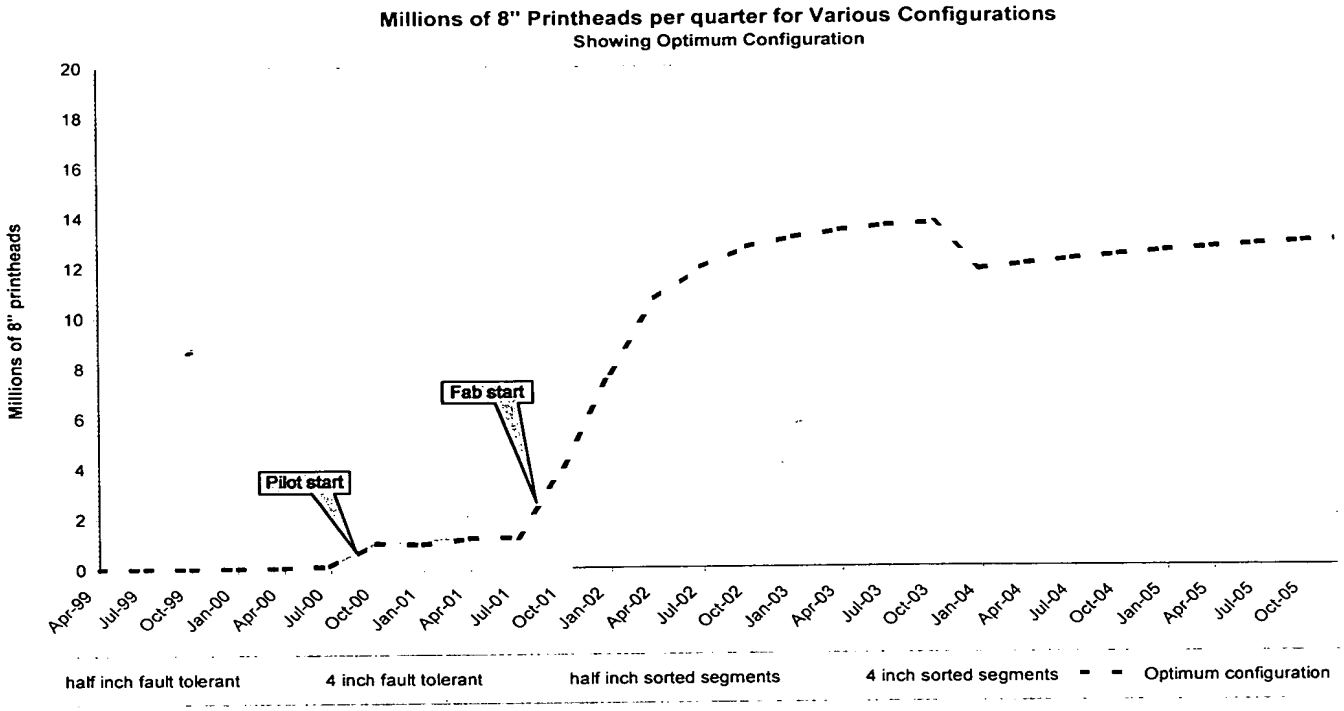
VOLUME

The manufacturing volume is predicted to go through three major phases:

1) An initial lab phase, where only around 100 wafers per month are processed, the yield are low to none, and the available number of printheads is very limited. Defect densities above 100 per cm^2 are assumed.

2) A pilot phase, where the number of wafers processed each month is 5,000. Defect densities are assumed to be between 10 per cm^2 and 100 per cm^2, necessitating the use of redundancy to achieve good printhead yields.

3) A full manufacturing phase, where 25,000 wafers are processed each month. Here defect densities are expected to fall from around 8 per cm^2 to 0.1 per cm^2 over the course of five years. During this phase, it is more cost effective *not* to use redundancy.



Early Product Mix

The market focus during the pilot phase and the main production phase should be different. During the pilot phase, markets which are relatively insensitive to print-head cost should be targeted. These include:

1) Wide format printing (machine prices around $10,000)

2) Network color printing (machine prices between $3,000 to $7,000)

3) Digital commercial printing (machine prices $100,000 and up)

4) Photo finishing (machine prices between $5,000 and $20,000)

During the main manufacturing phase, high volume markets can also be targeted.

MEMJET PROTOTYPE F. RICATION

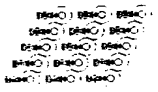
Before an integrated CMOS + MEMS prototype is made, we recommend the fabrication of a MEMS only prototype. The MEMS prototype can be made very faithfully to a full print head, with nearly identical actuator and nozzle structure. The main limitation of a MEMS only prototype is that the number of nozzles is limited, as a separate bond pad is required for each nozzle.

The prototype described here has only 15 nozzles per chip. The behavior of a few groups of 5 nozzles is a near perfect model of the entire chip performance, as the fluidic, thermal, electrical, acoustic, or mechanical coupling between 5 nozzle groups is extremely small.

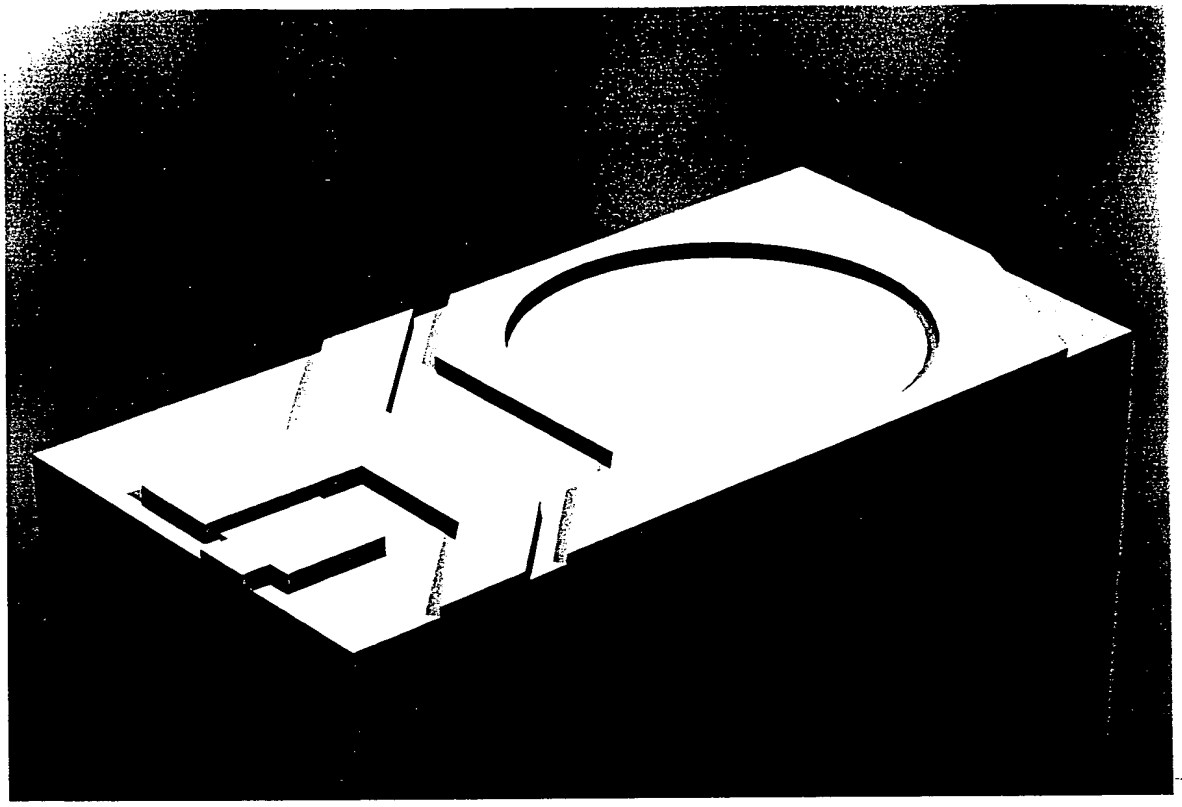
A chip layout with 15 nozzles is shown below. This chip is 3 mm x 3 mm, and is replicated on a 1.2 x 1.2 cm mask set. The mask set contains 10 variants of the precise nozzle dimensions, plus various test structures. The mask also contains a test printhead with around 3000 nozzles. As there is no CMOS on the chip, only some of the nozzles are 'wired up'. This chip is intended to test mechanical aspects of handling and packaging.

The mask has been laid out using 'Tanner Tools', and is available as Tanner or GDSII files.

The subsequent pages show the process steps, and the mask for a single nozzle unit cell.



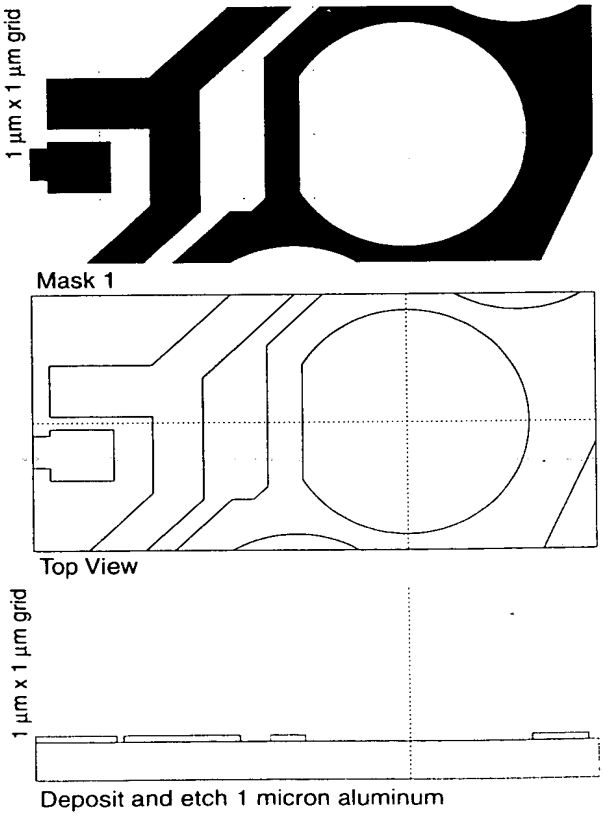
1) 1 Micron Aluminum



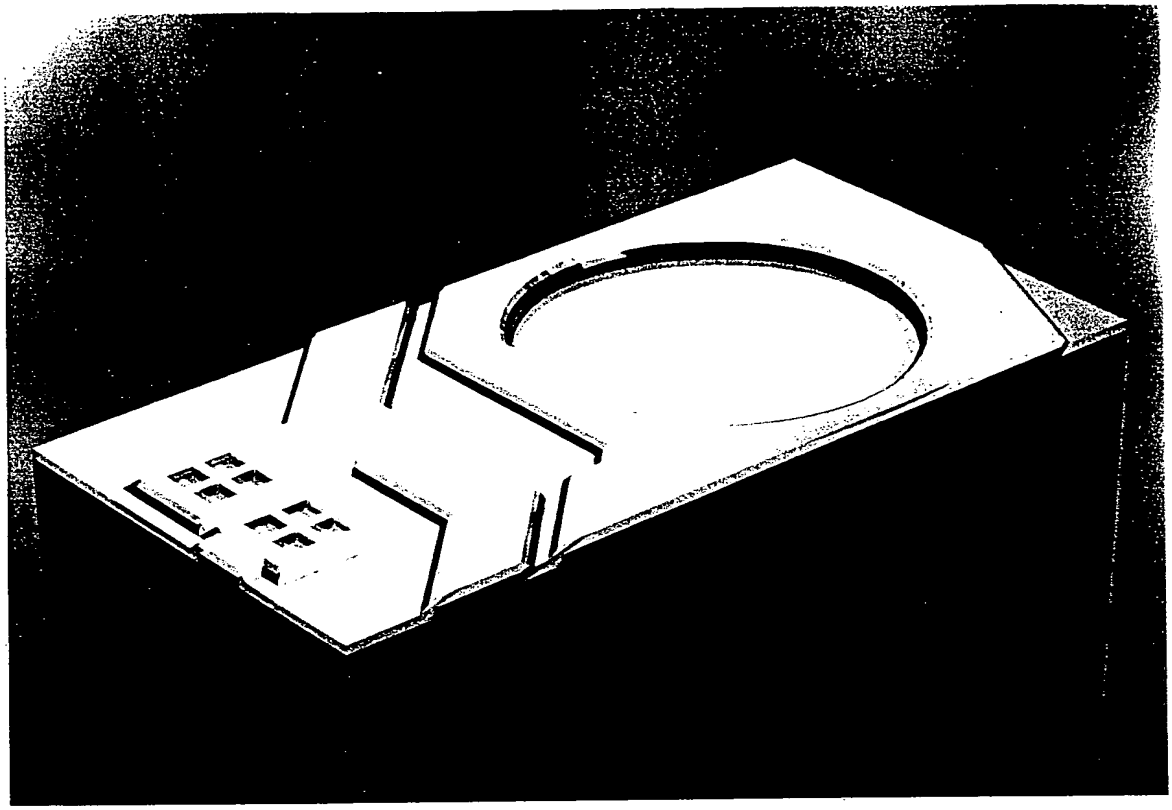
PROCESS DETAILS

One micron of aluminum is deposited and etched using Mask 1. This mask includes the electrodes to the actuator, the bond pads, and the wiring between these items. It is possible to replace the aluminum with TiN wiring and bond pads. However, that would diverge further from the CMOS + MEMS design, and add process risks. The region around the nozzle chamber is on Metal1 for a 1P2M CMOS + MEMS process, while the electrodes are on metal 2.

Layer thickness	1 micron (nominal)
Thickness variation	± 50%
Linewidth	3 microns
CD Accuracy	± 0.2 microns
Alignment accuracy	Mechanical
Align to	Wafer flat
Production process	CMOS M1 and M2
Production mask	Contains wiring



2) 1 Micron PECVD Nitride

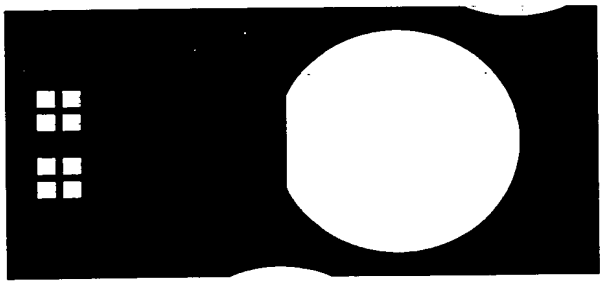


PROCESS DETAILS

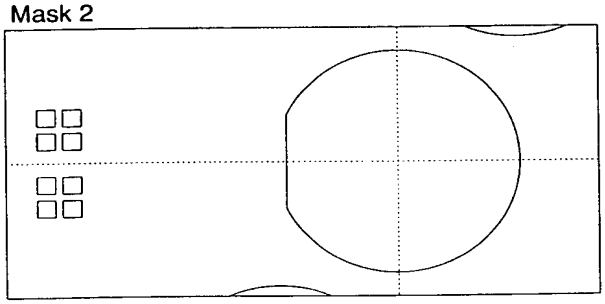
One micron of PECVD silicon nitride is deposited and etched using Mask 2. This mask includes the vias from the aluminum to the first TiN layer, and some relatively minor fluid control aspects. For a CMOS + MEMS process, this is the passivation layer, and will typically be 0.5 microns of glass followed by 0.5 microns of silicon nitride.

A pure nitride passivation layer is preferable, to prevent ions from the ink from diffusing through the glass.

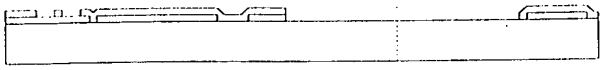
Layer thickness	1 micron
Thickness variation	± 20%
Linewidth	1 micron
CD Accuracy	± 0.2 microns
Alignment accuracy	± 0.2 microns
Align to	Metal 1
Production process	CMOS passivation
Production mask	Same



Mask 2

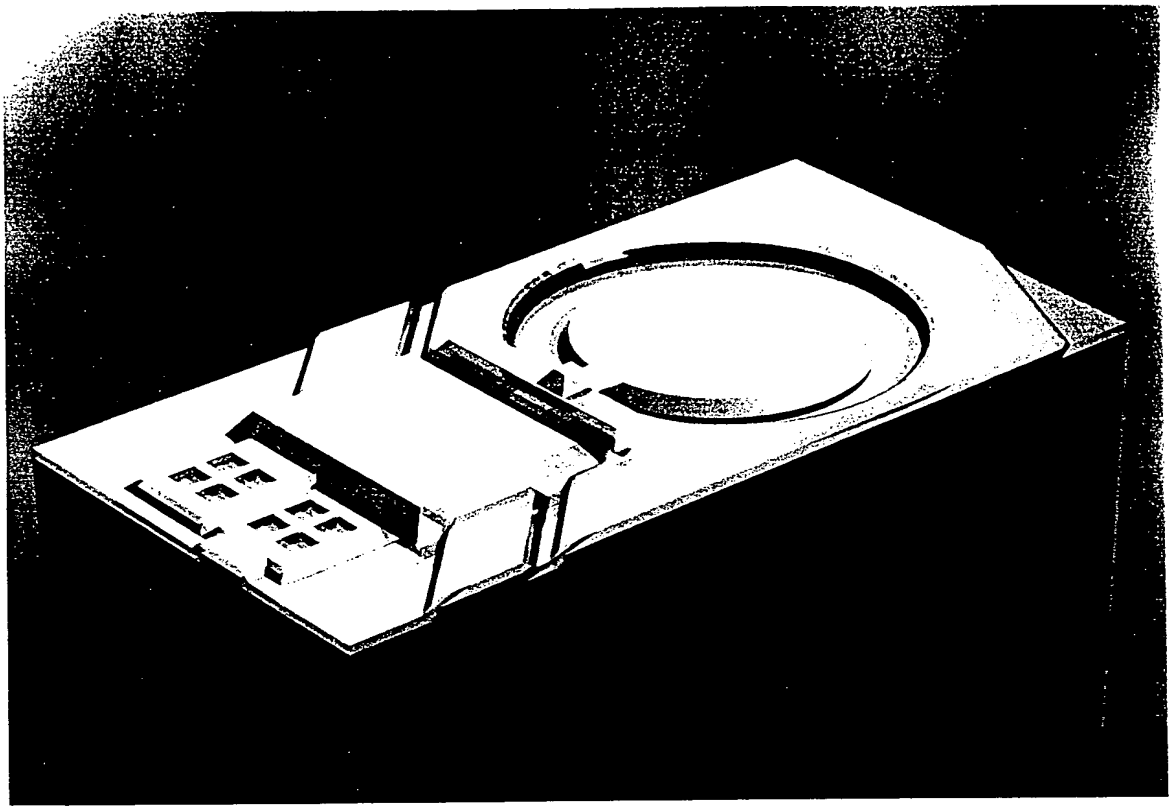


Top View



Deposit and etch 1micron PECVD $Si_xN_yH_z$

3) 1.5 Microns Sacrificial Polyimide

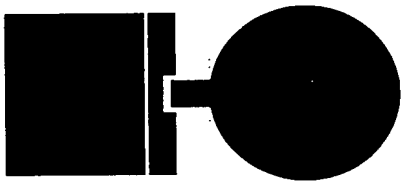


PROCESS DETAILS

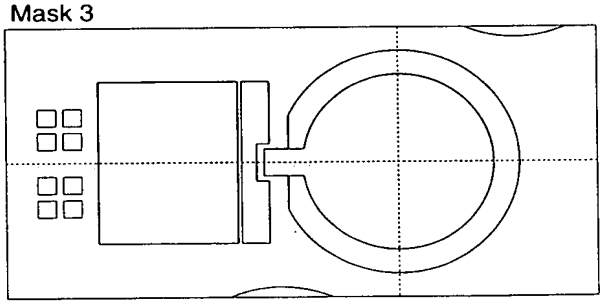
1.5 microns of spin-on photosensitive polyimide is deposited and exposed using UV light to Mask 3. The polyimide is then developed and hardbaked. 1.5 microns is the final thickness - around 3 microns of liquid is spun on, depending upon shrinkage.

The polyimide is sacrificial, so there is a wide range of alternative materials which can be used, such as glass or aluminum. Photosensitive polyimide simplifies the processing, as it eliminates deposition, etching, and resist stripping steps.

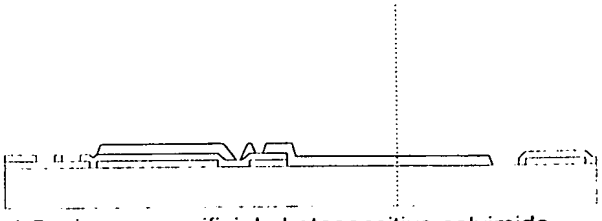
Layer thickness	1.5 microns
Thickness variation	± 10%
Polyimide sidewalls	approx. 60°
Linewidth	1.25 micron
CD Accuracy	± 0.15 microns
Alignment accuracy	± 0.15 microns
Align to	Metal 1
Production process	Same
Production mask	Same



Mask 3

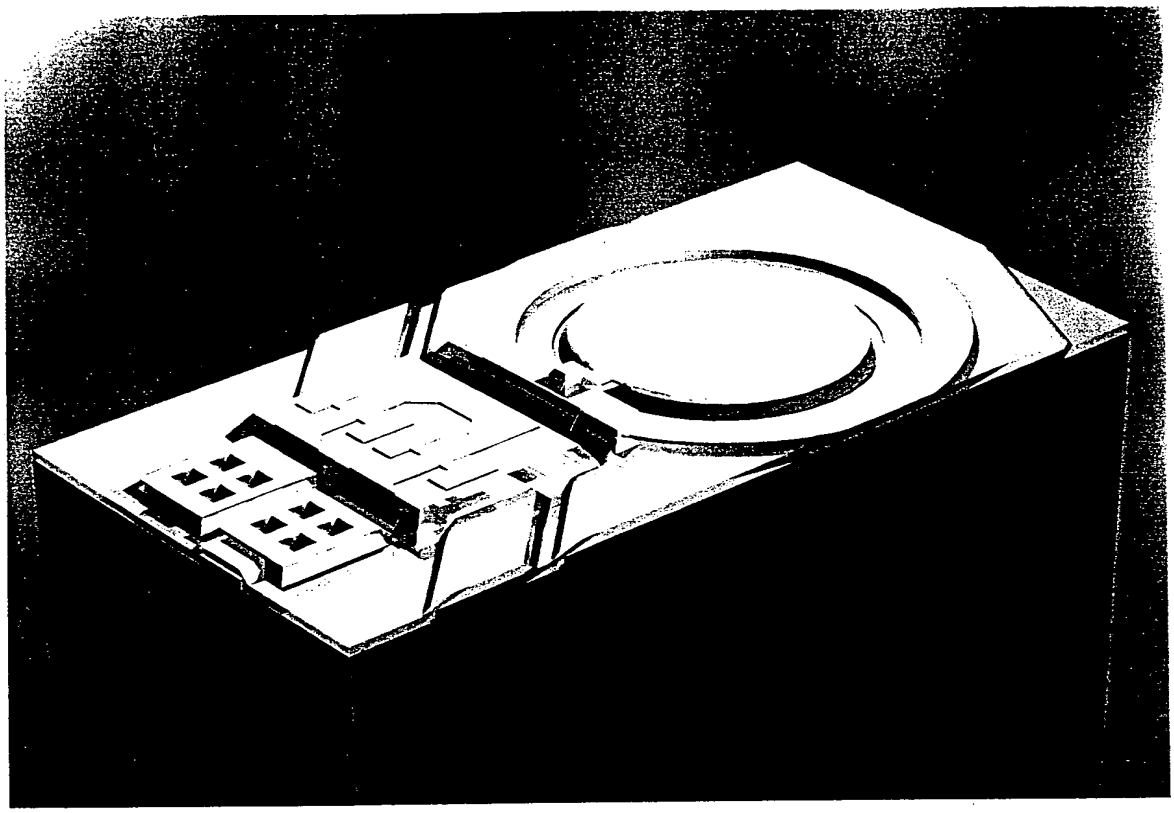


Top View



1.5 microns sacrificial photosensitive polyimide

4) Sputter 0.2 Microns TiN or (Ti,Al)N



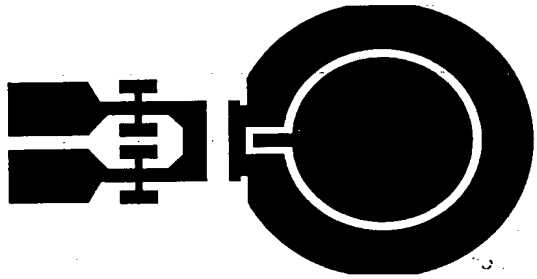
PROCESS DETAILS

0.2 microns of magnetron sputtered titanium nitride is deposited at 300 °C and etched using Mask 4. This layer contains the actuator and part of the paddle.

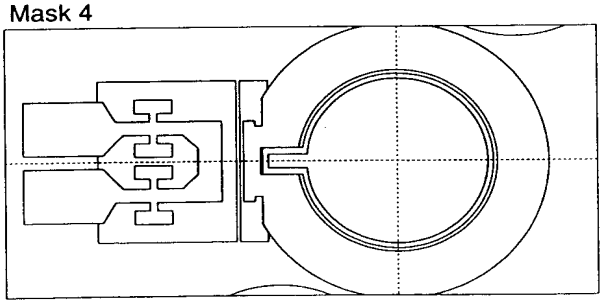
In production, the resistivity of this layer of TiN should be consistent to within a few percent over the wafer.

(Ti,Al)N is preferred to TiN for high efficiency operation, as it resists oxidation at higher temperatures.

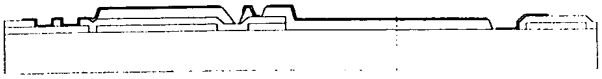
Layer thickness	0.2 microns
Thickness variation	± 5%
Linewidth	1 micron
CD Accuracy	± 0.15 microns
Alignment accuracy	± 0.1 microns
Align to	Metal 1
Production process	Same
Production mask	Same



Mask 4

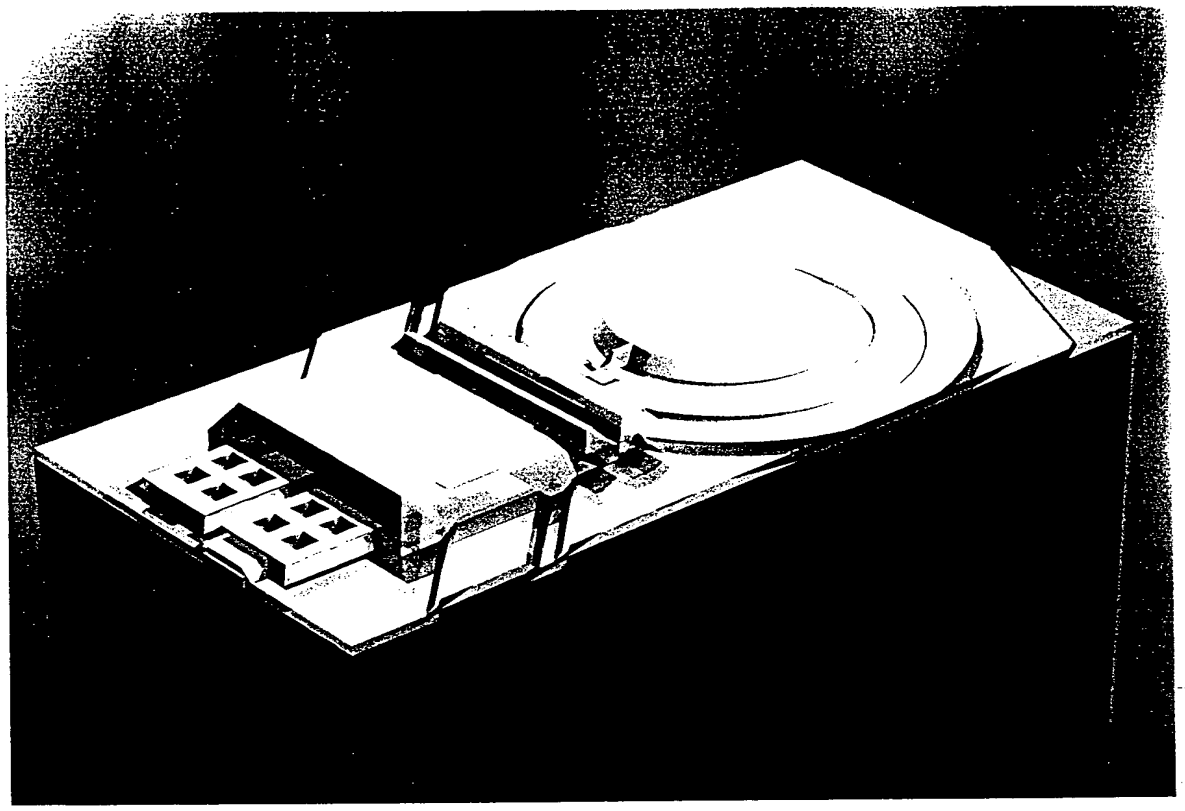


Top View



0.2 microns TiN sputtered at 300 degrees C

5) 1.5 Microns Sacrificial Polyimide



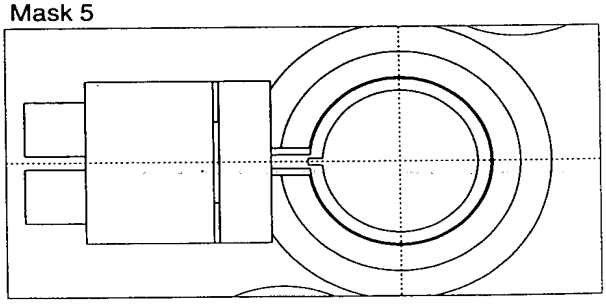
PROCESS DETAILS

This step is identical to step 3.

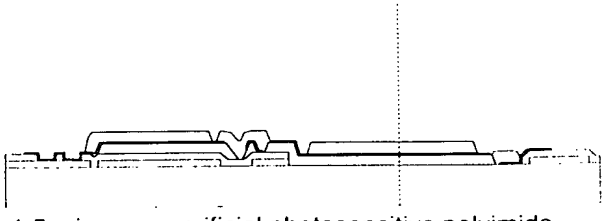
1.5 microns of spin-on photosensitive polyimide is deposited and exposed using UV light to Mask 5. The polyimide is then developed and hardbaked. 1.5 microns is the final thickness - spin on around 3 microns depending upon shrinkage. The thickness determines the gap between the actuator and compensator TiN layers, so has an effect on the amount that the actuator bends.

Layer thickness	1.5 microns
Thickness variation	± 10%
Polyimide sidewalls	approx. 60°
Linewidth	1 micron
CD Accuracy	± 0.5 microns
Alignment accuracy	± 0.4 microns
Align to	Metal 1
Production process	Same
Production mask	Same

Mask 5

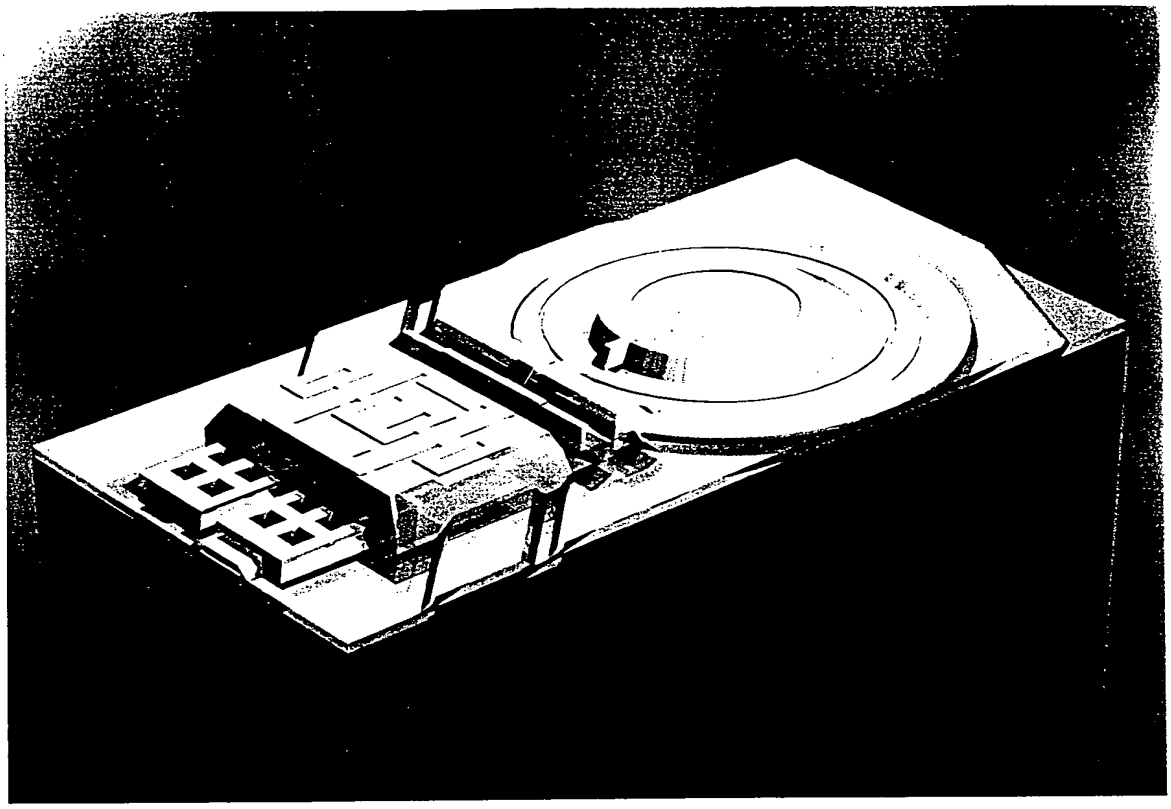


Top View



1.5 microns sacrificial photosensitive polyimide

6) 0.2 Microns Sputtered TiN



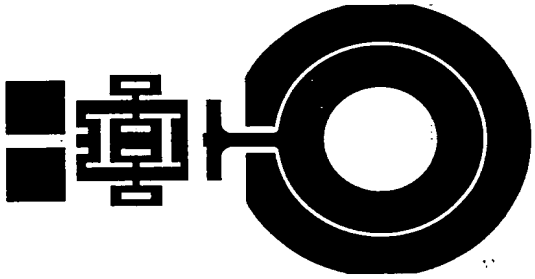
PROCESS DETAILS

This step may be identical to step 4. However, there is no efficiency advantage in the use of (Ti,Al)N, as this layer is not part of the thermal actuator. Either TiN or (Ti,Al)N may be used equivalently.

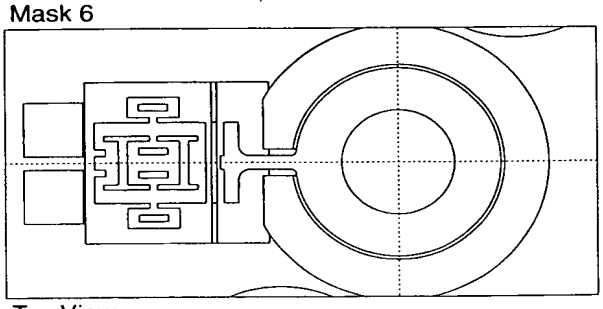
Deposit 0.2 microns of magnetron sputtered titanium nitride, at 300 °C. The TiN is etched using Mask 6.

This layer is not electrically connected, and is used purely as a mechanical component.

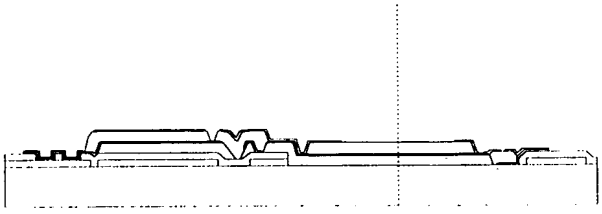
Layer thickness	0.2 microns
Thickness variation	± 5%
Linewidth	1 micron
CD Accuracy	± 0.15 microns
Alignment accuracy	± 0.15 microns
Align to	TiN 1
Production process	Same
Production mask	Same



Mask 6

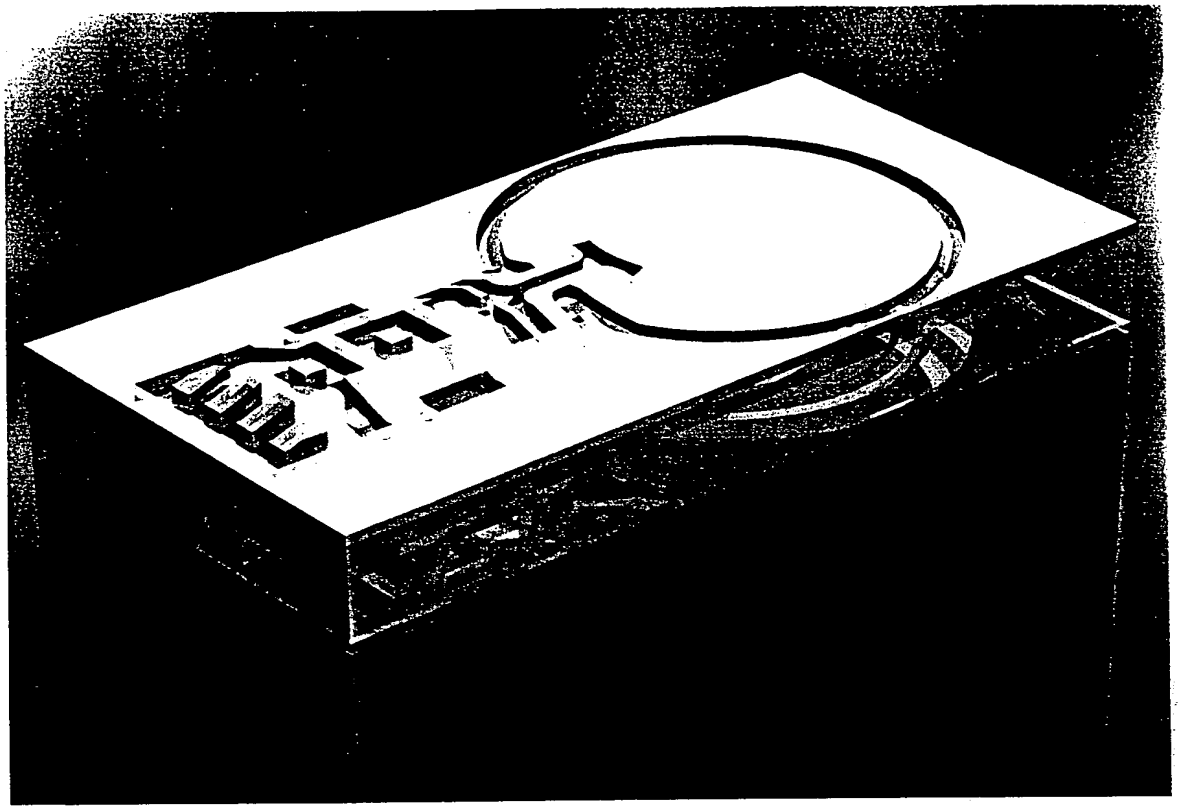


Top View



0.2 microns TiN sputtered at 300 degrees C

7) 8 Microns Sacrificial Polyimide, Al Mask

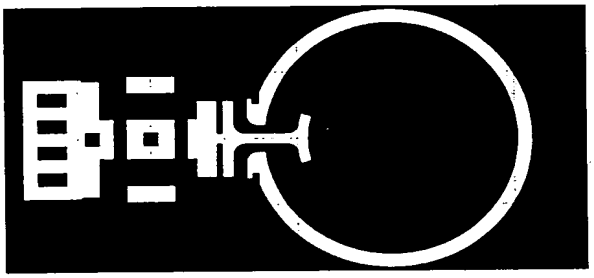


PROCESS DETAILS

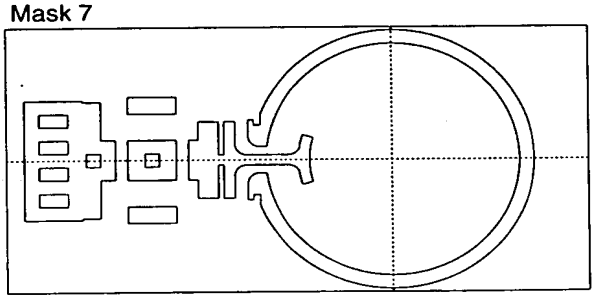
8 microns of standard polyimide is spun on and hardbaked. This thickness determines the height to the nozzle chamber roof. As long as this height is above a certain distance (determined by drop break-off characteristics), then the actual height is of little significance. As this polyimide layer is not photosensitive, it may be a filled layer to obtain a lower coefficient of thermal expansion.

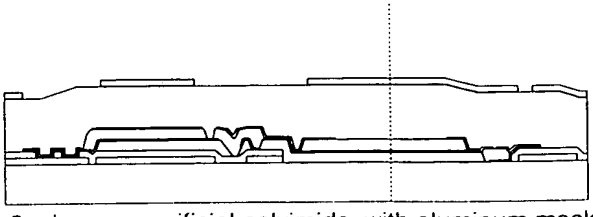
A 50 nm aluminum hard mask is deposited. One micron of resist is spun on and exposed to Mask 7. The Al hardmask is etched.

Layer thickness	8 microns
Thickness variation	+ 4, - 0.5 microns
Linewidth	2 microns
CD Accuracy	± 0.5 microns
Alignment accuracy	± 0.1 microns
Align to	TiN 1
Production process	Same
Production mask	Depends on etch



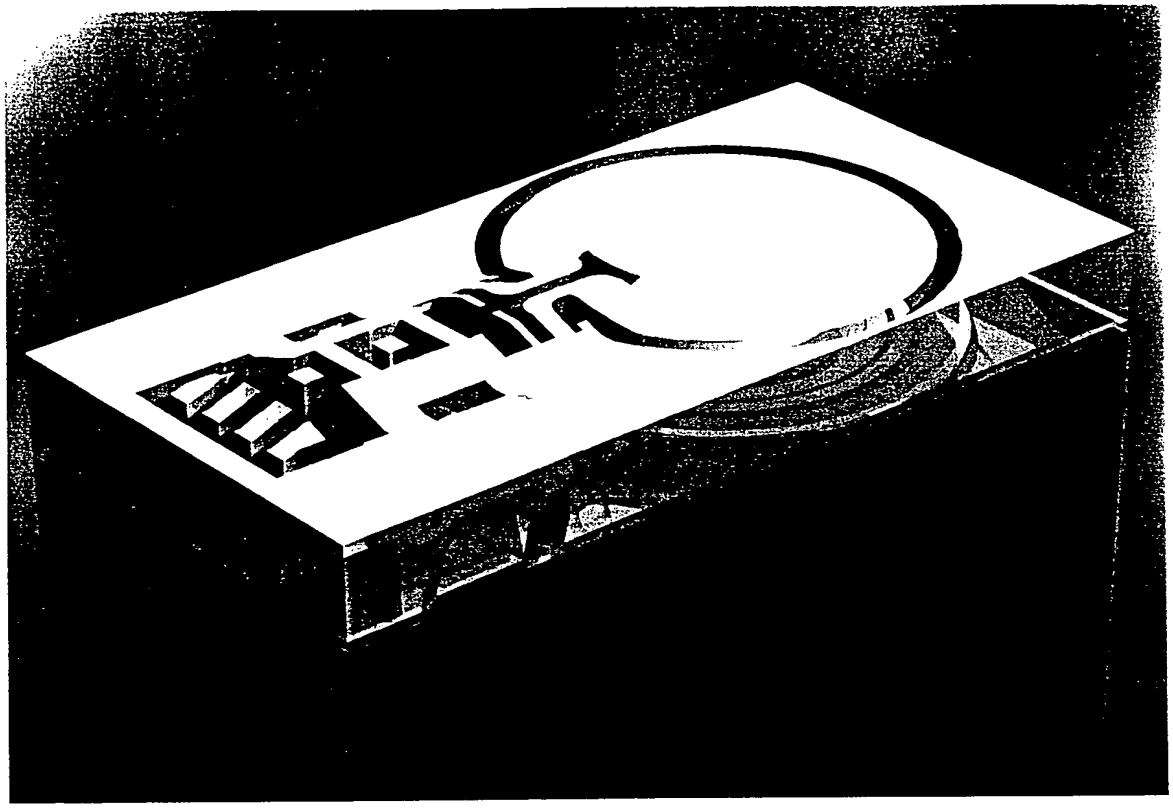
Mask 7





8 microns sacrificial polyimide, with aluminum mask

8) Etch Polyimide using Oxygen Plasma

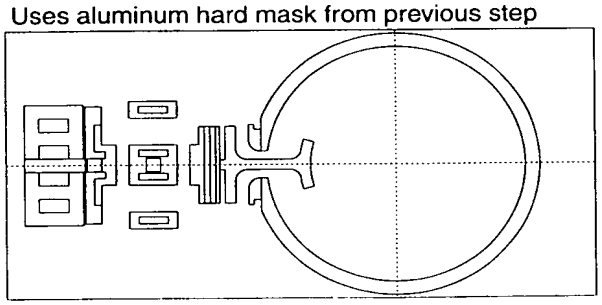


PROCESS DETAILS

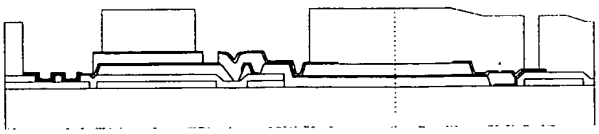
Sacrificial polyimide is anisotropically plasma etched. The sidewall angle should be better than 80 degrees. The mask design shown for Mask 7 is for 90 degree sidewalls, and should be modified to suit the etch process if the etch process used varies more than ± 3 degrees from vertical.

Etch depth	11 microns
Etch stop	TiN, Si_3N_4
Sidewall angle	90 degrees
Angle accuracy	± 3 degrees
Alignment accuracy	± 0.2 microns
Align to	TiN 2
Production process	Same

Uses aluminum hard mask from previous step

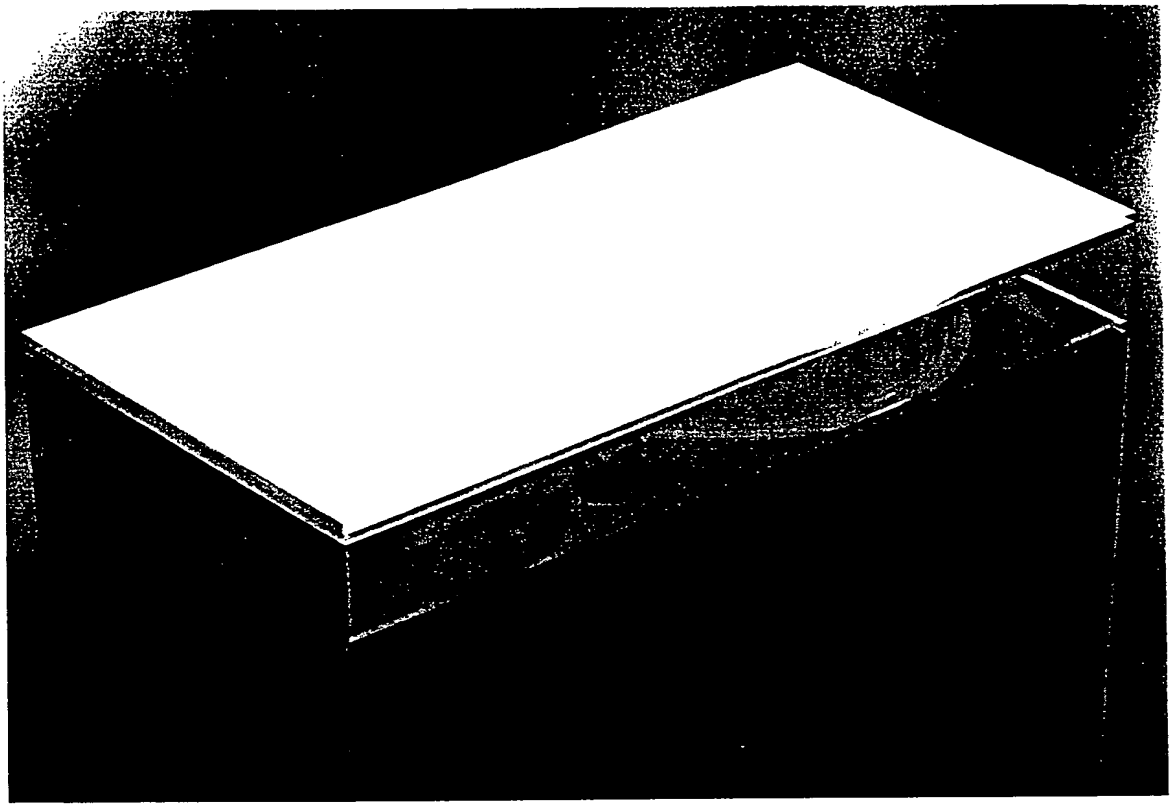


Top View



Etch sacrificial polyimide using oxygen plasma

9) 1 Micron Conformal Silicon Nitride



PROCESS DETAILS

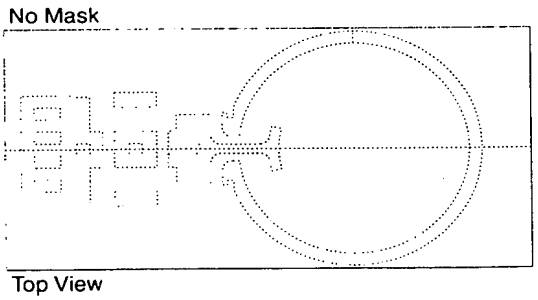
1 micron of PECVD silicon nitride is deposited at 300 °C. This fills the channels formed in the previous PS polyimide layer, forming the nozzle chamber.

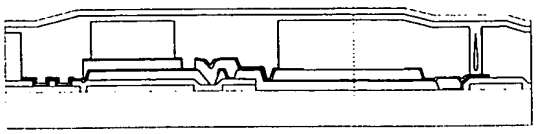
In the cross section, some areas appear to be large solid areas of nitride. These are actually 2 micron thick slots viewed side on.

This layer is not particularly critical. The major requirement is good adhesion to TiN. Enclosed vacuoles should not cause problems.

The nitride deposition is followed by 1 micron of polyimide, which is hardbaked.

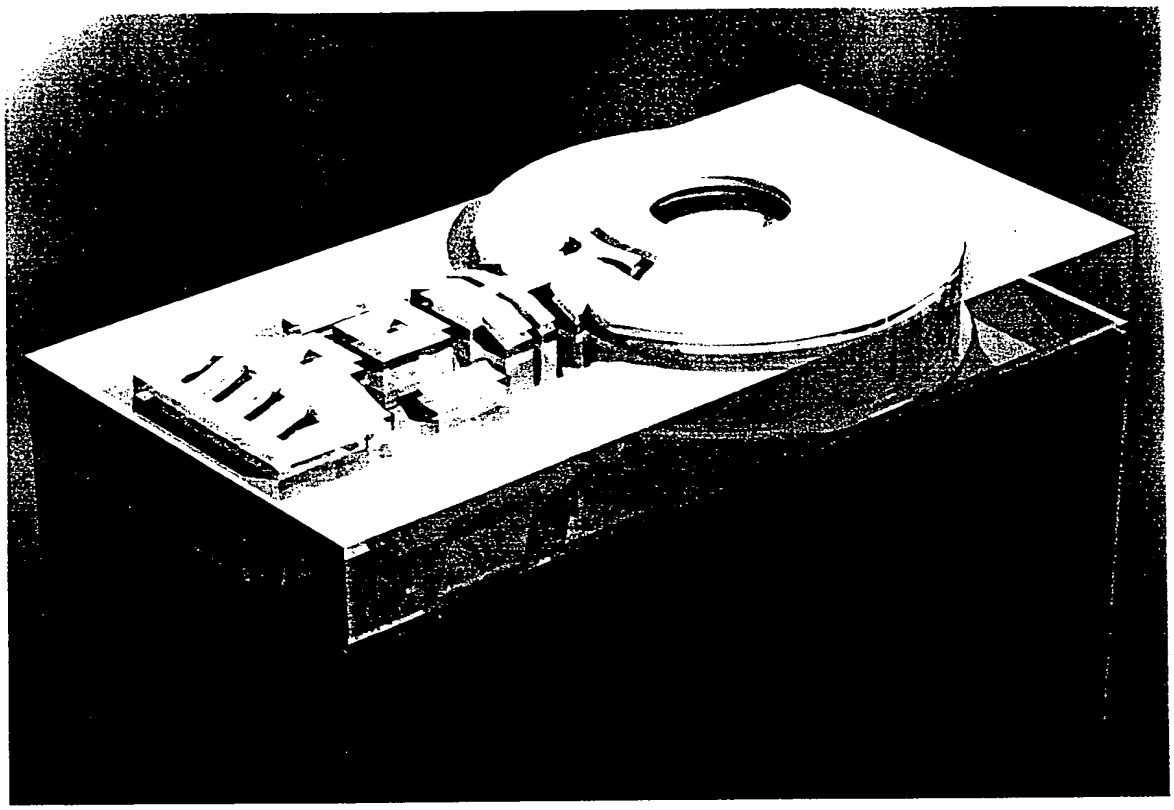
Si_3N_4 thickness	1 micron
Thickness variation	+/- 25%
Polyimide thickness	1 micron
Thickness variation	+/- 25%
Production process	Same





1 micron conformal PECVD $Si_xN_yH_z$, 1 micron polyimide

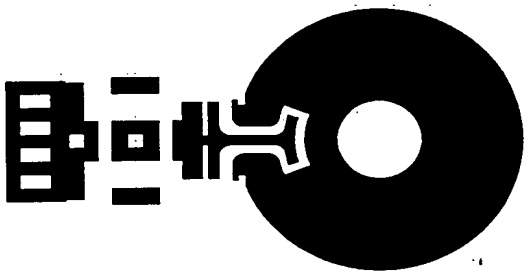
10) Etch Polyimide and Nitride



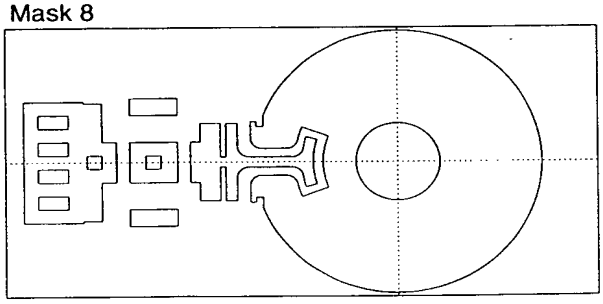
PROCESS DETAILS

The polyimide is etched down to nitride using Mask 8. The nitride is then etched down to Sac 3 polyimide using the Sac 4 polyimide as a mask.

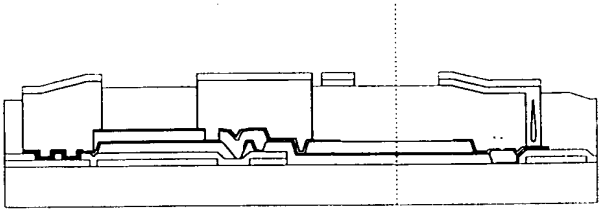
PI Etch depth	1.5 micron
Etch variation	± 10%
Nitride Etch depth	1.5 micron
Etch variation	± 10%
Linewidth	1 micron
CD Accuracy	± 0.15 microns
Alignment accuracy	± 0.1 microns
Align to	TiN 2
Production process	Same
Production mask	Same



Mask 8

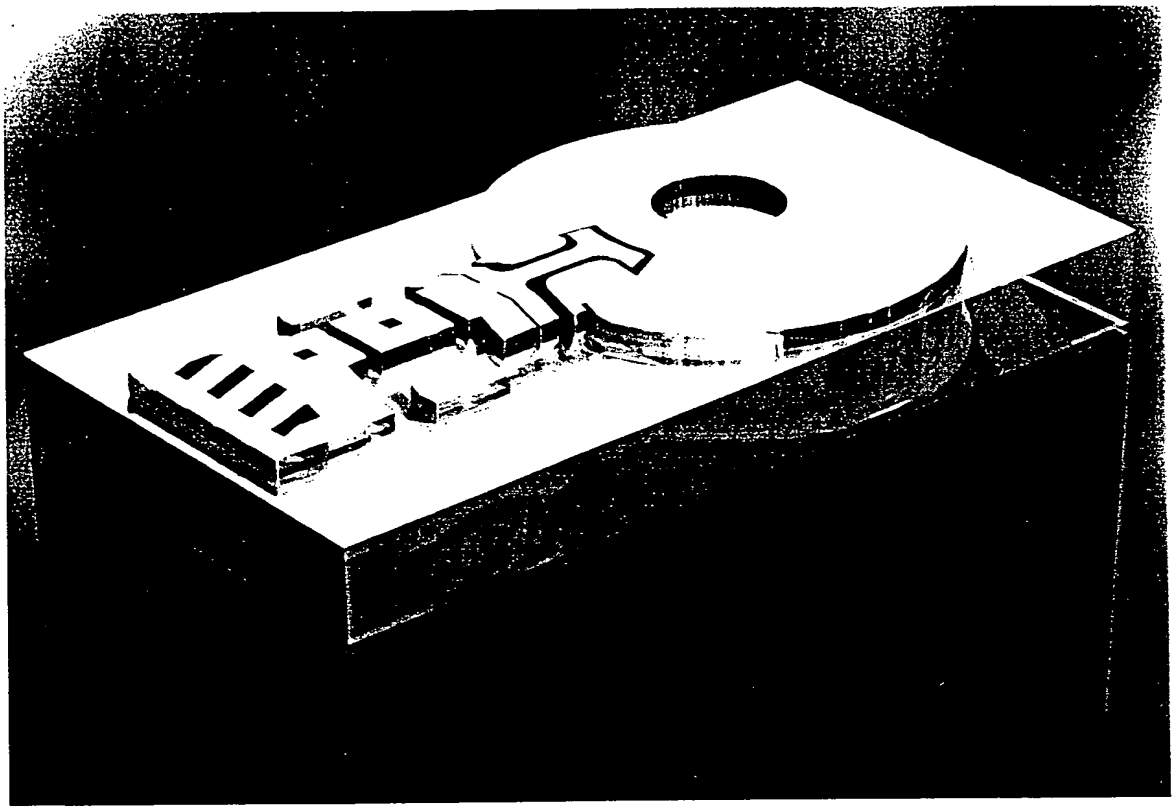


Top View



Etch of Sac 4 polyimide and nozzle roof $Si_xN_yH_z$

11) Deposit 0.25 Microns of PECVD Nitride

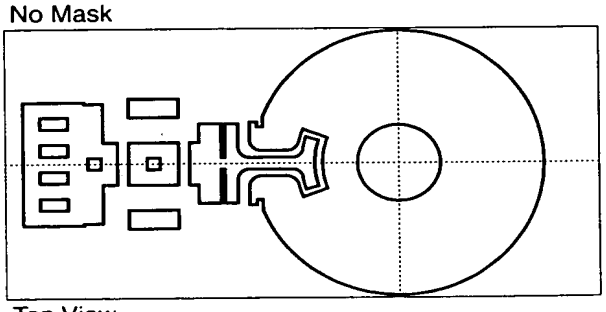


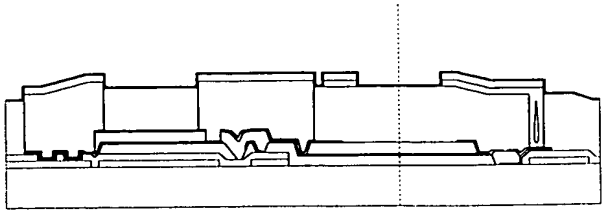
PROCESS DETAILS

0.25 microns of conformal PECVD silicon nitride is deposited at 300°C.

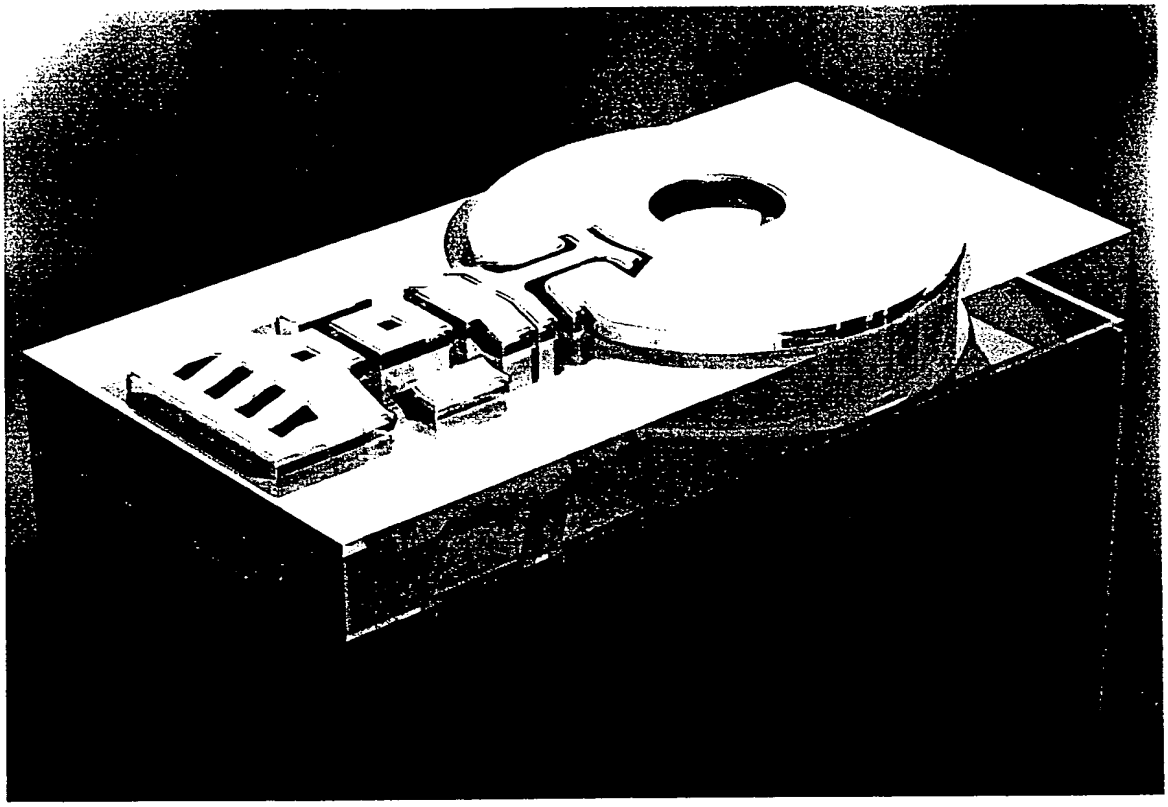
This layer forms the nozzle rims, using a 'sidewall spacer' like process. The thickness is not particularly critical, and could be substantially thinner if desired, as there is insignificant fluidic pressure acting on the rim.

Si_3N_4 thickness	nominal 0.25 microns
Thickness variation	± 25%
Production process	Same





12) Anisotropic Etch of Nitride

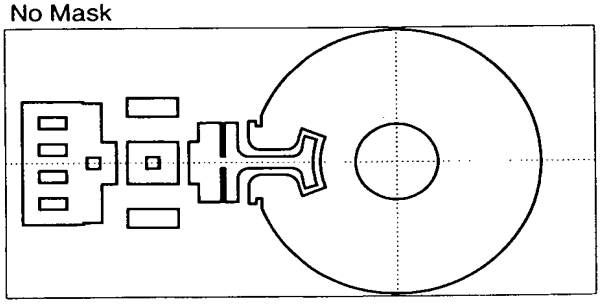


PROCESS DETAILS

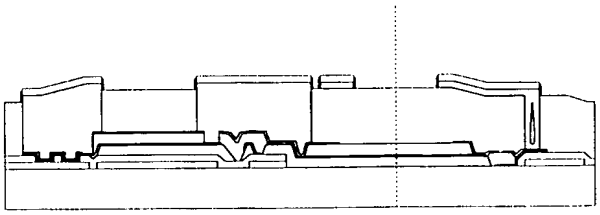
The nozzle rim nitride is anisotropically plasma etched. The etch can be timed, as etch depth is not critical. Substantial overetch is required to ensure that only vertical nitride walls remain, and that nitride over sloping topography is completely removed.

Etch depth	0.5 microns
Depth variation	± 20%
Production process	Same
Production mask	None

No Mask

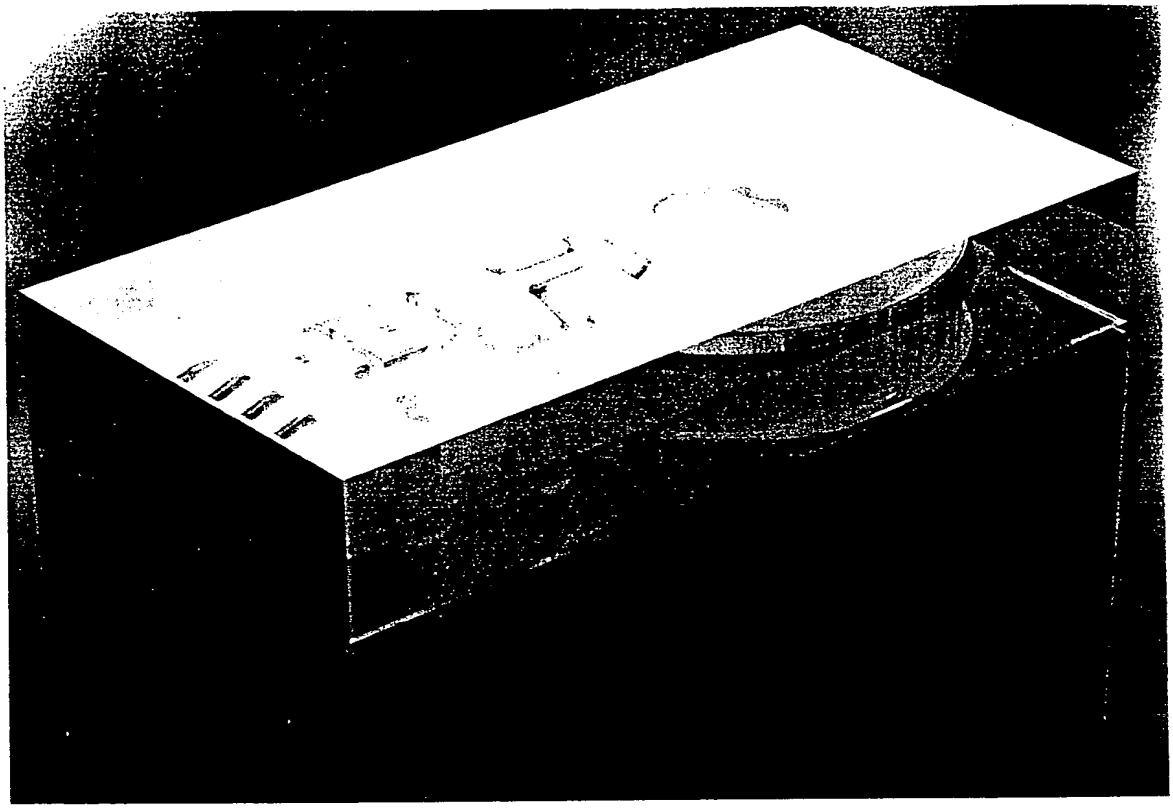


Top View



0.5 micron anisotropic 'sidewall' etch of $Si_xN_yH_z$

13) 4 Microns of Softbaked Resist



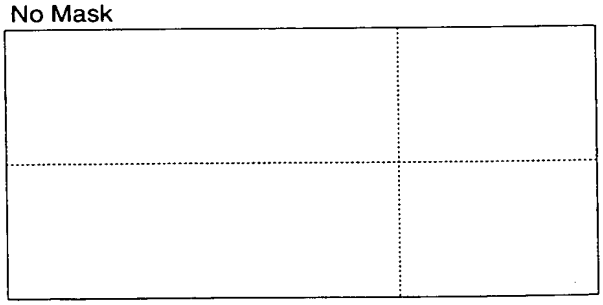
PROCESS DETAILS

Spin on 4 microns of resist and softbake.

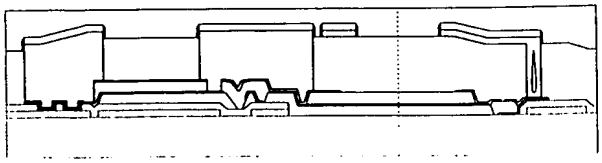
This resist layer is to protect the front side of the wafer during backetch. The resist thickness is to cover the topography of the MEMS devices, and thereby allow a vacuum chuck to be used.

Resist thickness	4 microns nominal
Thickness variation	+50% - 25%
Production process	Same
Production mask	None

No Mask

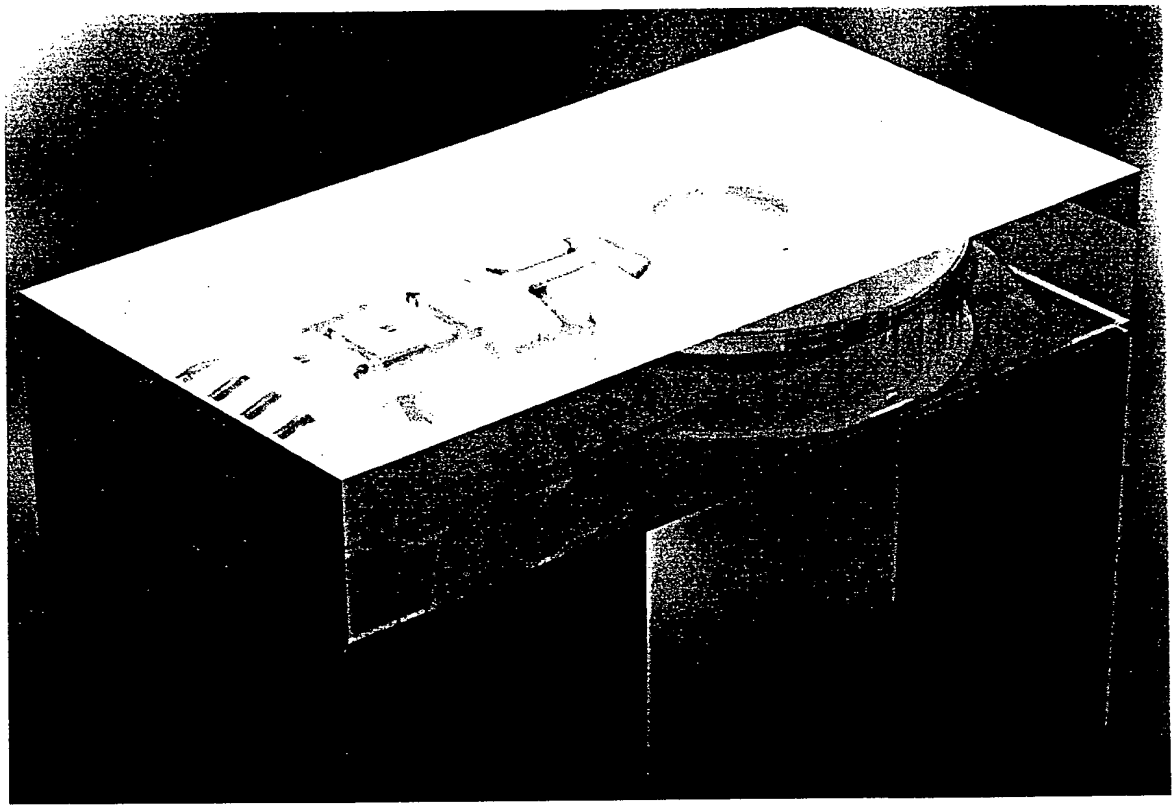


Top View



4 microns softbaked resist as a protective layer

14) Back-etch using Bosch Process

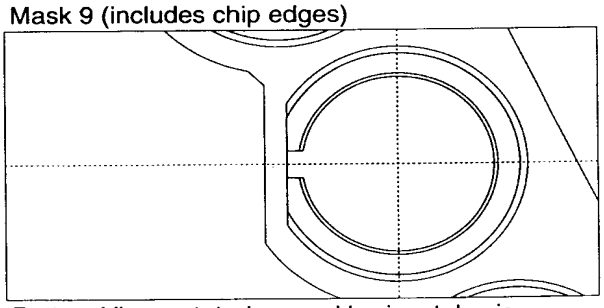


PROCESS DETAILS

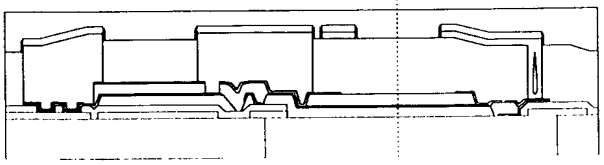
The wafer is thinned to 300 microns (to reduce back-etch time), and 3 microns of resist on the back-side of the wafer is exposed to Mask 10. Alignment is to metal 1 on the front side of the wafer. This alignment can be achieved using an IR microscope attachment to the wafer aligner. The wafer is then placed on a platter and etched to a depth of 330 microns (allowing 10% overetch) using the deep silicon etch "Bosch process". This process is available on plasma etchers from Alcatel, PlasmaTherm, and Surface Technology Systems.

Etch depth	330 micron
Etch variation	± 10%
Linewidth	40 microns
CD Accuracy	± 2 microns
Alignment accuracy	± 2 microns
Align to	Metal 1
Production process	Same
Production mask	Full wafer

Mask 9 (includes chip edges)

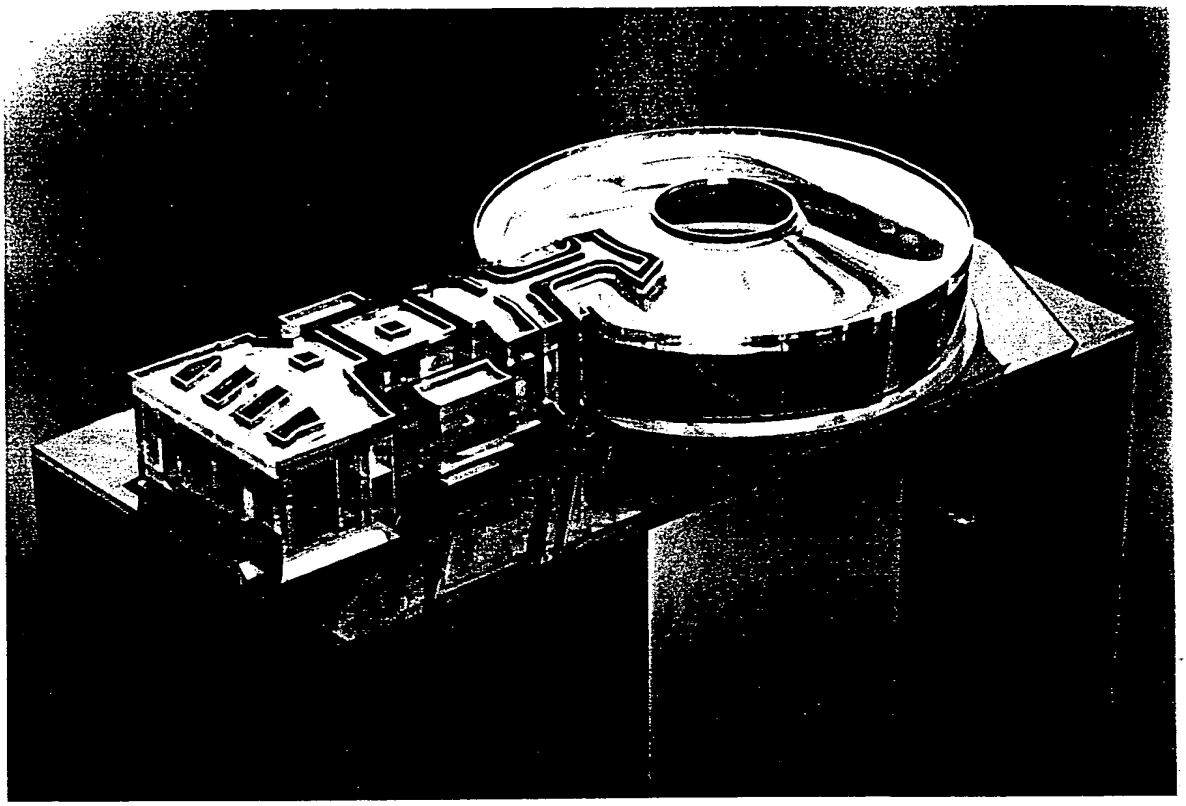


Bottom View, rotated around horizontal axis



Back-etch through wafer using Bosch process

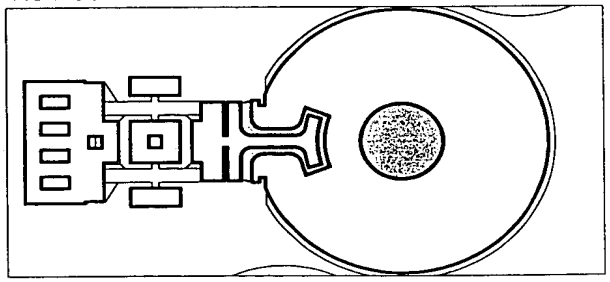
15) Strip all Sacrificial Material

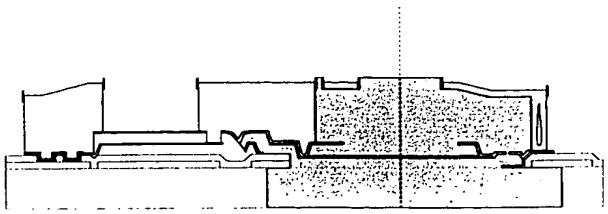


PROCESS DETAILS

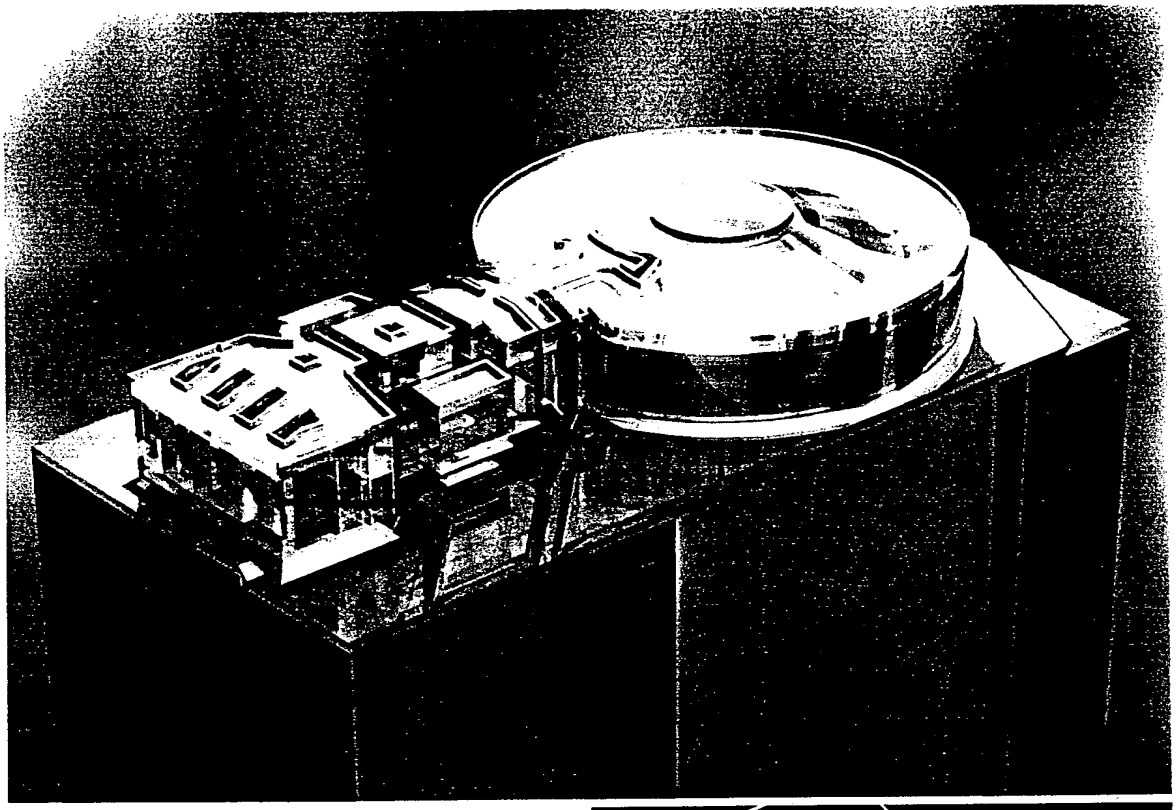
The chips were diced by the previous Bosch process back-etch. However, the wafer is still held together by 11 microns of polyimide. The wafers must now be turned over. This can be done by placing a tray over the wafer on the platter, and turning the whole assembly (platter, wafer and tray) over while maintaining light pressure. The platter is then removed, and the wafer (still in the tray) is placed in the oxygen plasma chamber.

All of the sacrificial polyimide is ashed in an oxygen plasma.





16) Package, Bond, and Prime.

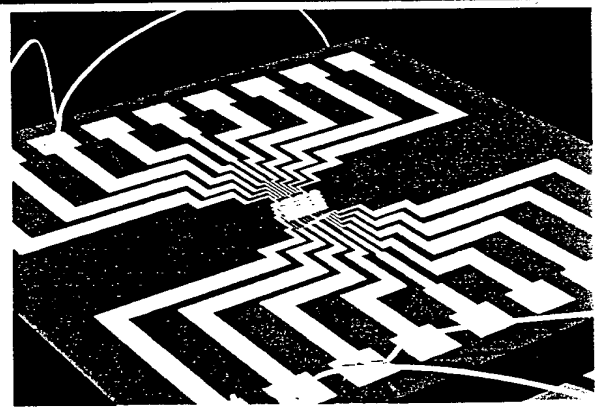


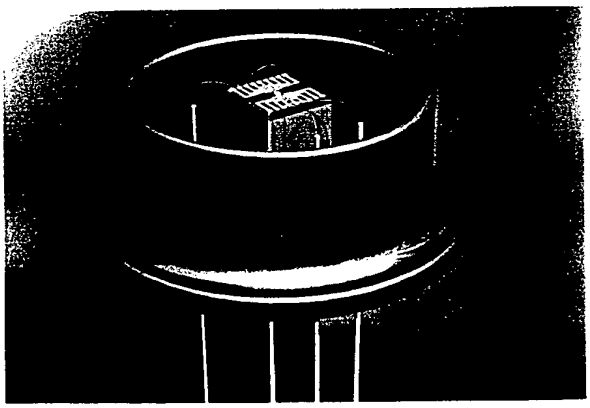
PROCESS DETAILS

Glue the chip into a package with an ink inlet hole, for example, a pressure transducer package. The ink hose should include a 0.5 micron absolute filter to prevent contamination of the nozzles.

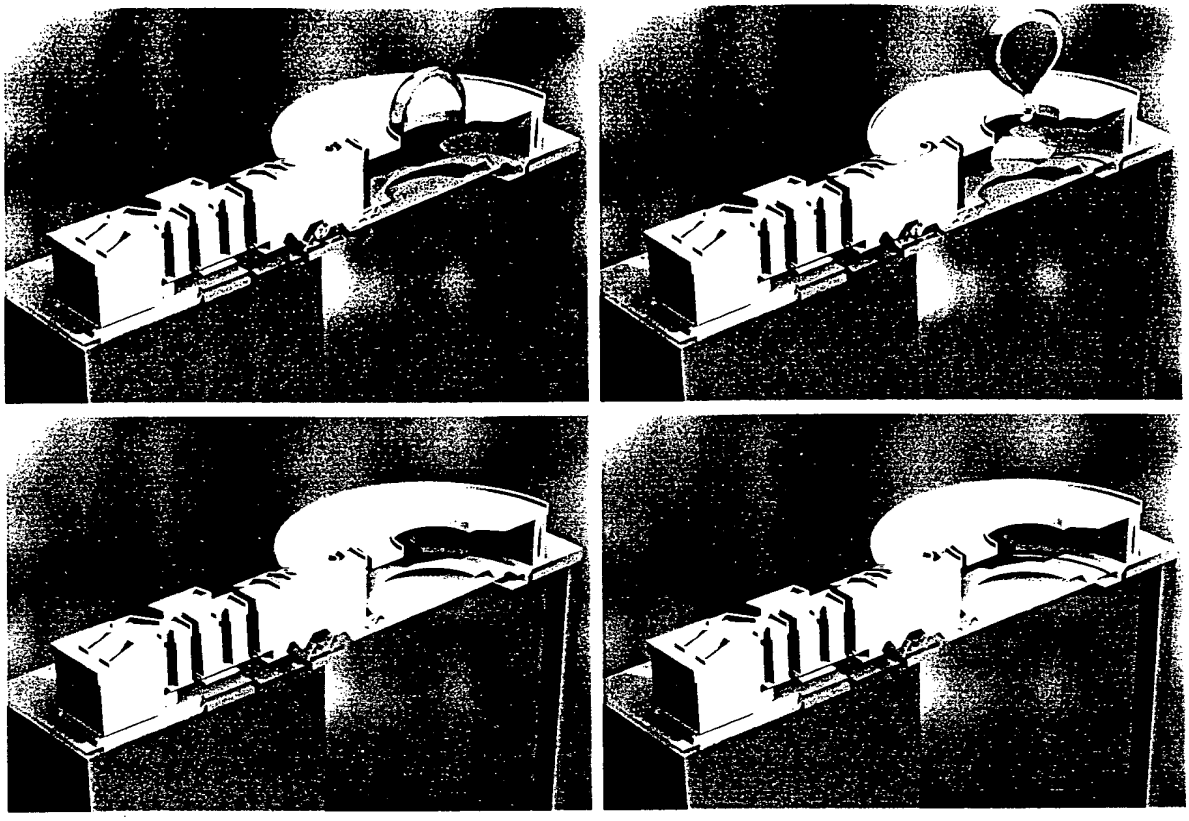
The prototype Memjet chips are 3 mm square, but the ink inlet hole region is only about 240 x 160 microns, in the center of the chip. Glue the chip into the package so that the chip ink inlet is over the hole in the package. This requires only 500 micron accuracy. Wire bond the 6 connections to nozzles to be tested.

Fill the packaged printhead under approx. 5 kPa ink pressure to prime it.





17) Test

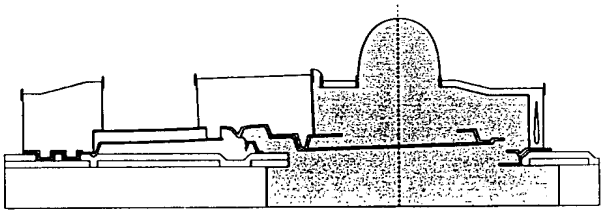


PROCESS DETAILS

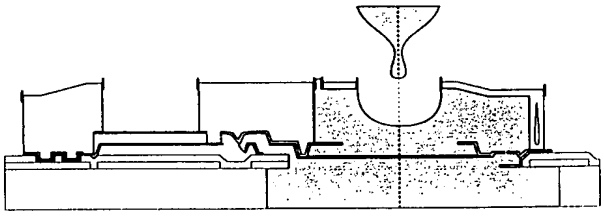
Testing of the inkjets is performed by a mixture of physical observation and simulation.

The primary physical test is to take high speed photomicrographs of the ink drops in flight. This is done with a special optical microscope, a high speed flash, a gated photomultiplier plate, and a high speed CCD camera. The images are stored on computer and carefully compared to the simulation results. When the simulation matches the experimental images to within a few percent, then the simulation can act as a 'microscope' able to see almost any aspect of the nozzle operation.

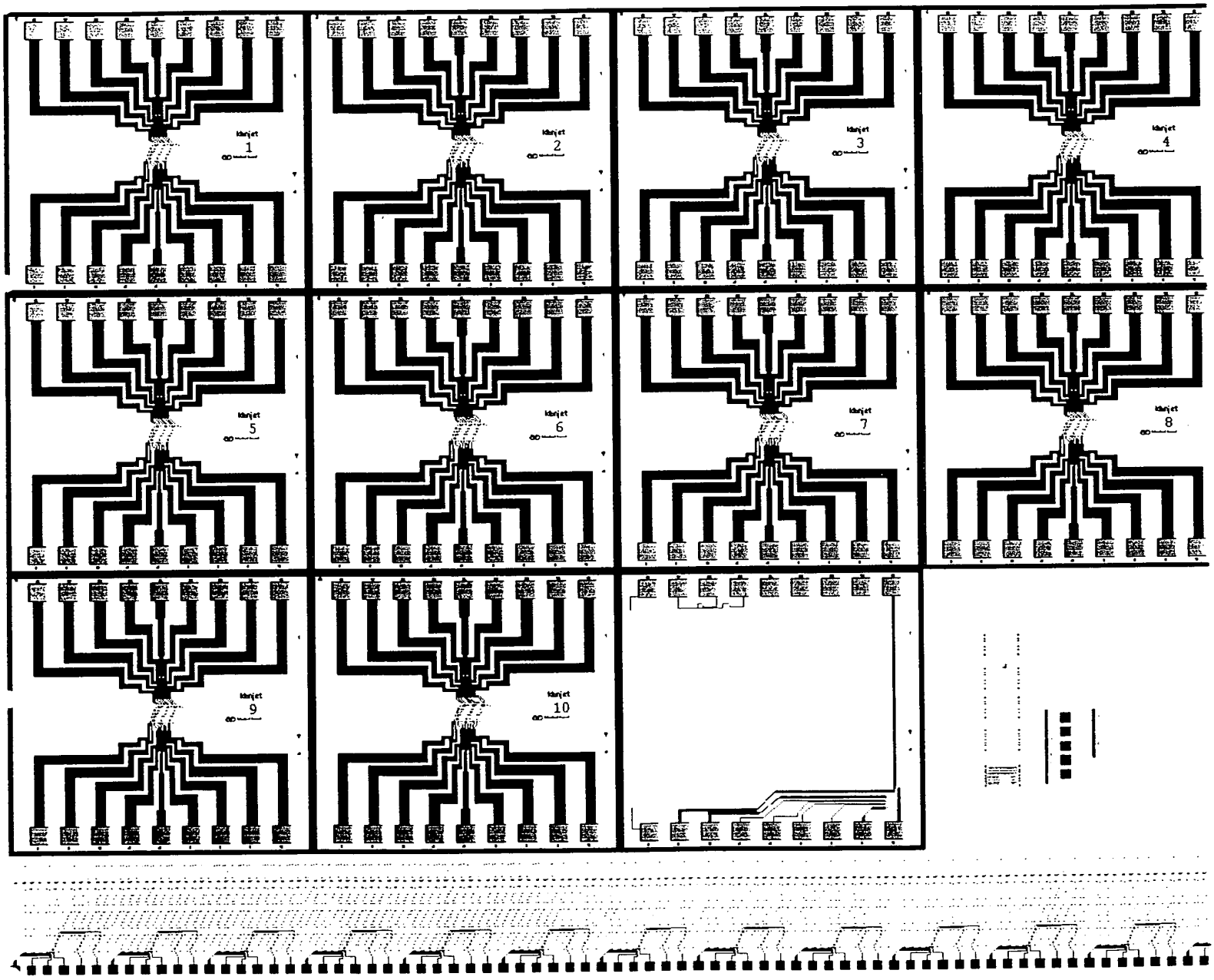
Silverbrook Research is establishing a testing laboratory that will be able to perform the necessary tests, and correlate these tests with extensive simulation results.



Actuate

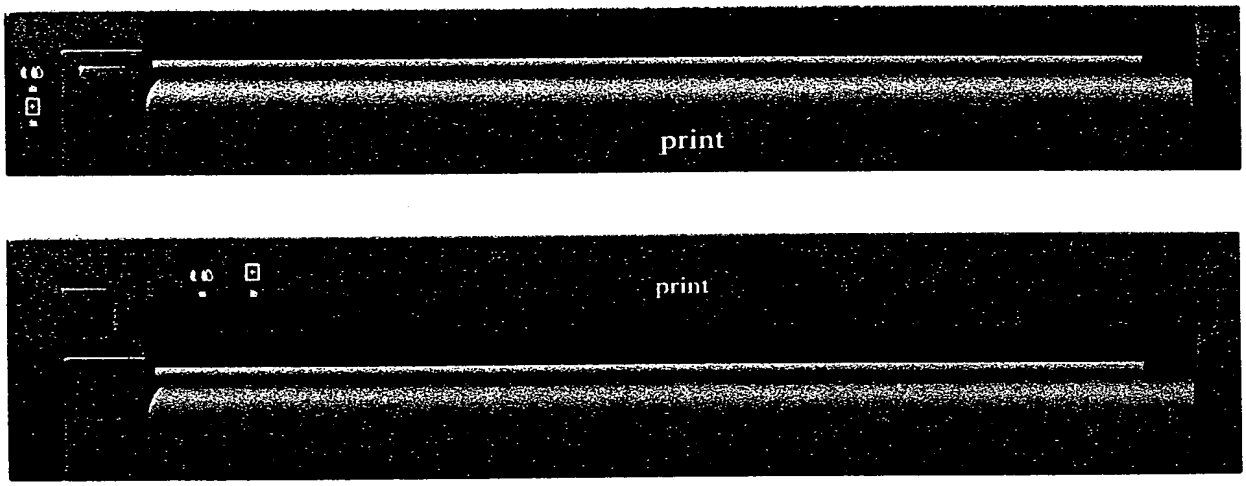


Return


kbnjet
1
kbnjet
2
kbnjet
3
kbnjet
4
kbnjet
5
kbnjet
6
kbnjet
7
kbnjet
8
kbnjet
9
kbnjet
10

CePrint
Product Design Description

draft version 1.1



Appendix B

Silverbrook Research Pty Ltd
393 Darling Street, Balmain
NSW 2041 Australia
Phone: +61 2 9818 6633
Fax: +61 2 9818 6711
Email: info@silverbrook.com.au

Document History

Version	Date	Authors	Details
1.1d	12 January 1999	Paul Lapstun Toby King Simon Walmsley	Expanded description of printer hardware. Added CCP internal timings. Added line sync for stepper motor.
1.0d	8 January 1999	Paul Lapstun Simon Walmsley Toby King Kia Silverbrook	Initial draft issue.

Contents

1 Introduction 1
1.1 Operational Overview 1
1.2 Document Summary 1
2 Product Specification 2
3 Memjet-Based Printing 6
4 Page Delivery Architecture 7
4.1 Page Image Sizes 7
4.2 Constraints 7
4.3 Page Rendering and Compression 8
4.4 Page Expansion and Printing 10
5 Printer Hardware 12
5.1 Overview 14
5.2 Printhead Assembly and Image Transfer Mechanism 17
5.3 Manufacturing Cost 23
6 Printer Control Protocol 27
6.1 Control and Status 27
6.2 Page Description 28
6.2.1 Page Structure 29
6.2.2 Page Description Format 29
6.2.3 Bi-level Black Layer Compression 31
6.2.4 Contone Layer Compression 37
7 Printer Controller 39
7.1 Printer Controller Architecture 39
7.2 Page Expansion and Printing 39
7.2.1 DMA Approach 42
7.2.2 EDRL Expander 42
7.2.3 JPEG Decoder 50
7.2.4 Halftoner/Compositor 50
7.3 Printhead Interface 55
7.3.1 Timing 56
7.4 Processor and Memory 56
7.4.1 Processor 56
7.4.2 DMA Controller 56
7.4.3 Program ROM 57
7.4.4 Rambus Interface 57
7.5 External Interfaces 57
7.5.1 Host Interface 57
7.5.2 Speaker Interface 57
7.5.3 Parallel Interface 57
7.5.4 Serial Interface 58
7.5.5 Inter-CCP Interface 59
7.5.6 JTAG Interface 59
8 Double-Sided Printing 60
8.1 Page Delivery and Distribution 60
8.2 Synchronized Printing 61

9 Memjet Printhead 62
9.1 The Structure of a Memjet Segment 62
9.1.1 Grouping of Nozzles Within a Segment 62
9.1.2 Load and Print Cycles 66
9.1.3 Feedback from a Segment 70
9.1.4 Preheat Cycle 70
9.1.5 Cleaning Cycle 70
9.1.6 Printhead Interface Summary 71
9.2 Making Memjet Printheads out of Segments 72
9.2.1 Loading Considerations 73
9.2.2 Printing Considerations 74
9.2.3 Feedback Considerations 74
9.2.4 Printhead Connection Summary 75
10 Memjet Printhead Interface 76
10.1 Block Diagram of Printhead Interface 77
10.2 LineSyncGen Unit 77
10.3 Memjet Interface 78
10.3.1 Connections to Printhead 79
10.3.2 Firing Pulse Duration 80
10.3.3 Dot Counts 81
10.3.4 Registers 82
10.3.5 Preheat and Cleaning Cycles 85
10.4 Line Loader/Format Unit 85
10.4.1 Buffers 87
10.4.2 Address Generation 88
10.4.3 Registers 90
10.5 Controlling a Print 90
11 Generic Printer Driver 91
11.1 Graphics and Imaging Model 91
11.2 Two-Layer Page Buffer 91
11.3 Compositing Model 92
11.4 Page Compression and Delivery 93
11.5 Banded Output 94
12 Windows 9x/NT/CE Printer Driver 95
12.1 Windows 9x/NT/CE Printing System 95
12.2 Windows 9x/NT/CE Graphics Device Interface (GDI) 96
12.3 Printer Driver Graphics DLL 96
12.3.1 Managing the Device Surface 96
12.3.2 Determining Contone Object Geometry 98
12.3.3 Rendering Text 98
12.3.4 Banded Output 99
13 References 100

1 Introduction

CePrint is a high-performance color printer which combines photographic-quality image reproduction with magazine-quality text reproduction. It utilizes an 8" page-width Memjet [14] printhead which produces 1600 dots per inch (dpi) bi-level CMYK (Cyan, Magenta, Yellow, blacK).

CePrint is conceived as an OEM part designed for inclusion primarily in consumer electronics (CE) devices. Intended markets include televisions, VCRs, PhotoCD players, DVD players, Hi-fi systems, Web/Internet terminals, computer monitors, and vehicle consoles. It features a low-profile front panel and provides user access to paper and ink via an ejecting tray. It operates in a horizontal orientation under domestic environmental conditions.

CePrint exists in single- and double-sided versions. The single-sided version prints 30 full-color A4 or Letter pages per minute. The double-sided version prints 60 full-color pages per minute (i.e. 30 sheets per minute). Although CePrint supports both paper sizes, it is configured at time of manufacture for a specific paper size.

1.1 Operational Overview

CePrint reproduces black text and graphics directly using bi-level black, and continuous-tone (contone) images and graphics using dithered bi-level CMYK. For practical purposes, CePrint supports a black resolution of 800 dpi, and a contone resolution of 267 pixels per inch (ppi).

CePrint is embedded in a CE device, and communicates with the CE device (host) processor via a relatively low-speed (1.5MBytes/s) connection. CePrint relies on the host processor to render each page to the level of contone pixels and black dots. The host processor compresses each rendered page to less than 3MB for sub-two-second delivery to the printer. CePrint decompresses and prints the page line by line at the speed of the Memjet printhead. CePrint contains sufficient buffer memory for two compressed pages (6MB), allowing it to print one page while receiving the next, but does not contain sufficient buffer memory for even a single uncompressed page (119MB).

The double-sided version of CePrint contains two printheads which operate in parallel. These printheads are fed by separate data paths, each of which replicates the logic found in the single-sided version of CePrint. The double-sided version has a correspondingly faster connection to the host processor (3MB/s).

1.2 Document Summary

This document begins by giving a summary product specification for CePrint (Section 2). It then describes Memjet-based printing in general (Section 3), and the page delivery architecture which dictates much of CePrint's design (Section 4).

It goes on to specify the design of the printer hardware (Section 5), the printer control protocol (Section 6), the printer controller chip (Section 7), double-sided printing (Section 8), the Memjet printhead and interface (Section 9 and Section 10), a generic host-based printer driver (Section 11), and a Windows 9x/NT/CE printer driver (Section 12).

The printer hardware section (Section 5) includes estimated manufacturing costs for CePrint and its ink cartridge consumable.

2 Product Specification

Table 1 gives a summary product specification of the single-sided and double-sided versions of the CePrint unit. Figure 1 and Figure 2 illustrate the two versions. Figure 3 and Figure 4 show the front panels of the two versions actual size.

Table 1. CePrint Specification

	single-sided version	double-sided version
dot pitch	1600 dpi	
paper	standard A4 / Letter	
paper tray capacity	150 sheets	
print speed	30 pages per minute	60 pages per minute, 30 sheets per minute
warm-up time	nil	
first print time	2-6 seconds	
subsequent prints	2 seconds / sheet	
color model	32-bit CMYK process color	
printable area	full page (full edge bleed)	
printhead	pagewidth Memjet with 54,400 nozzles	dual printheads
print method	self-cleaning transfer roller, titanium nitride (TiN) coated	dual transfer rollers
size (h × w × d)	40mm × 272mm × 416mm	60mm × 272mm × 416mm
weight	3kg (approx.)	4kg (approx.)
power supply	5V, 4A	5V, 8A
ink cartridge color capacity	650 pages at 15% coverage	
ink cartridge black capacity	900 pages at 15% coverage	
ink cartridge size (h × w × d)	21mm × 188mm × 38mm	

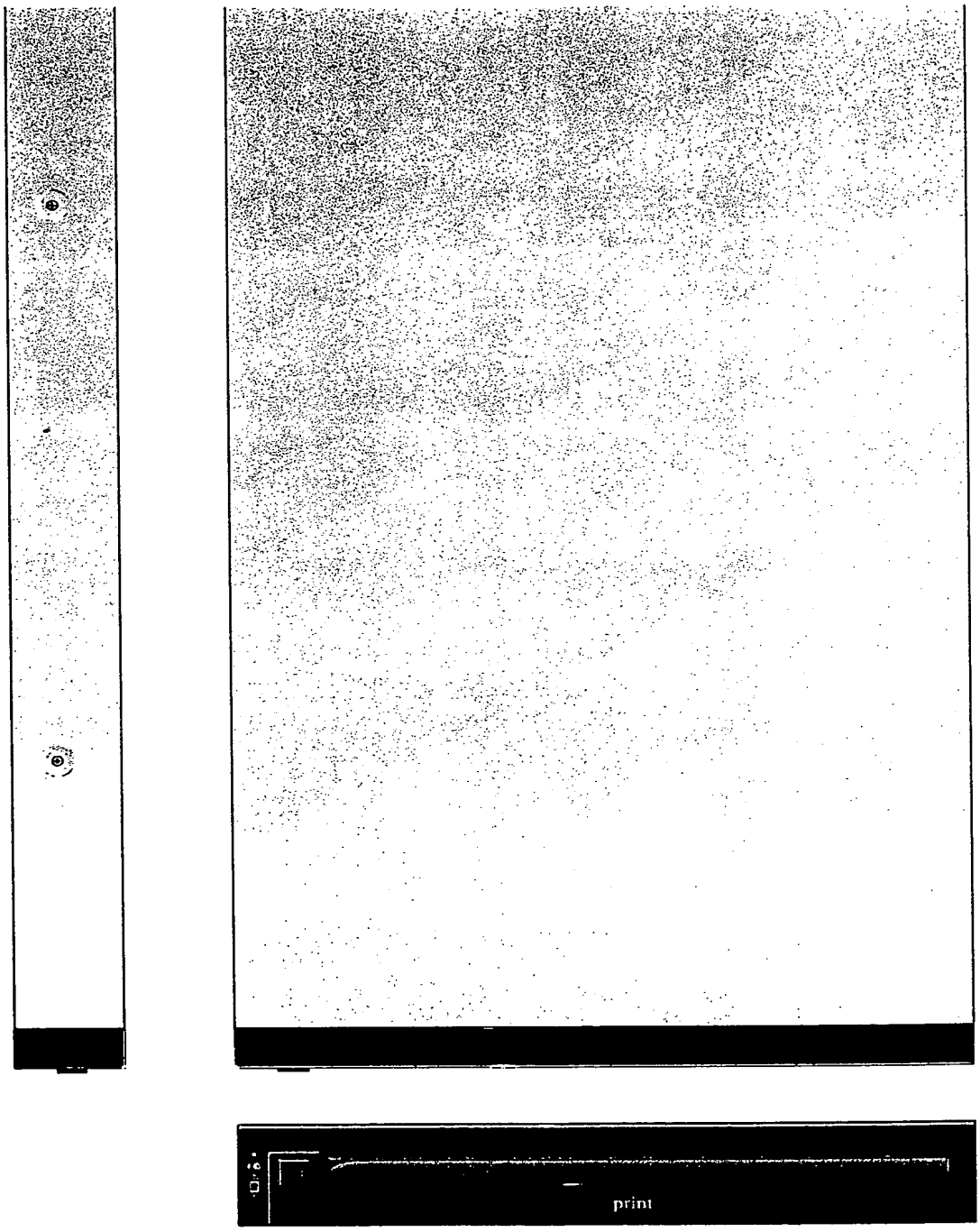


Figure 1. Single-sided version of CePrint (40mm h × 272mm w × 416mm d)

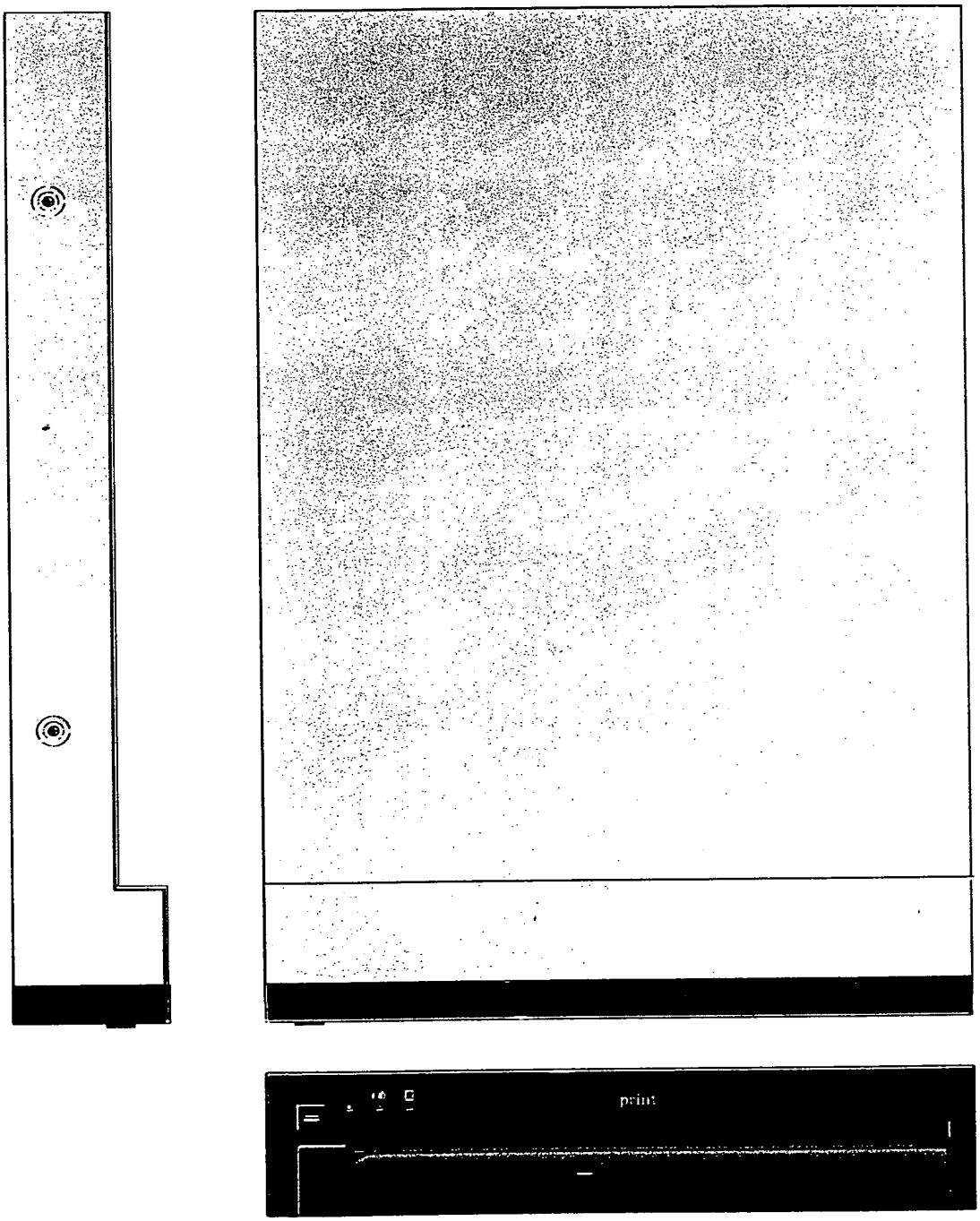


Figure 2. Double-sided version of CePrint (60mm h × 272mm w × 416mm d)

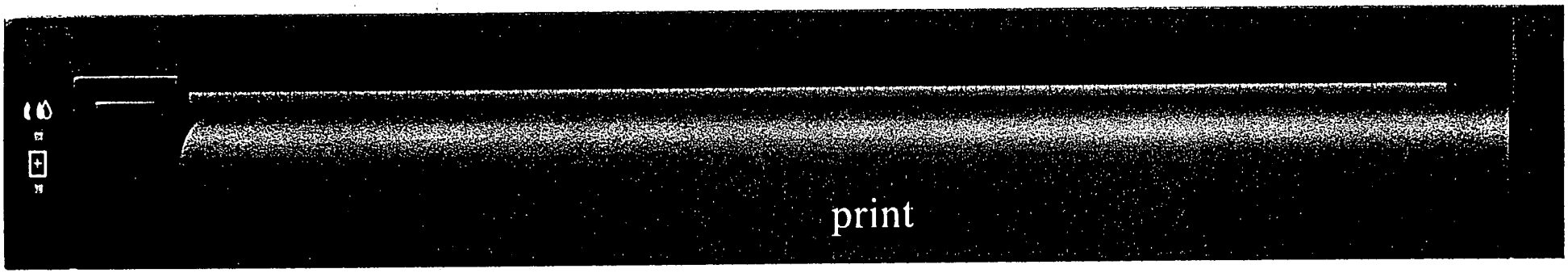


Figure 3. Front panel of single-sided version of CePrint OEM printer shown actual size (40mm h x 272mm w)

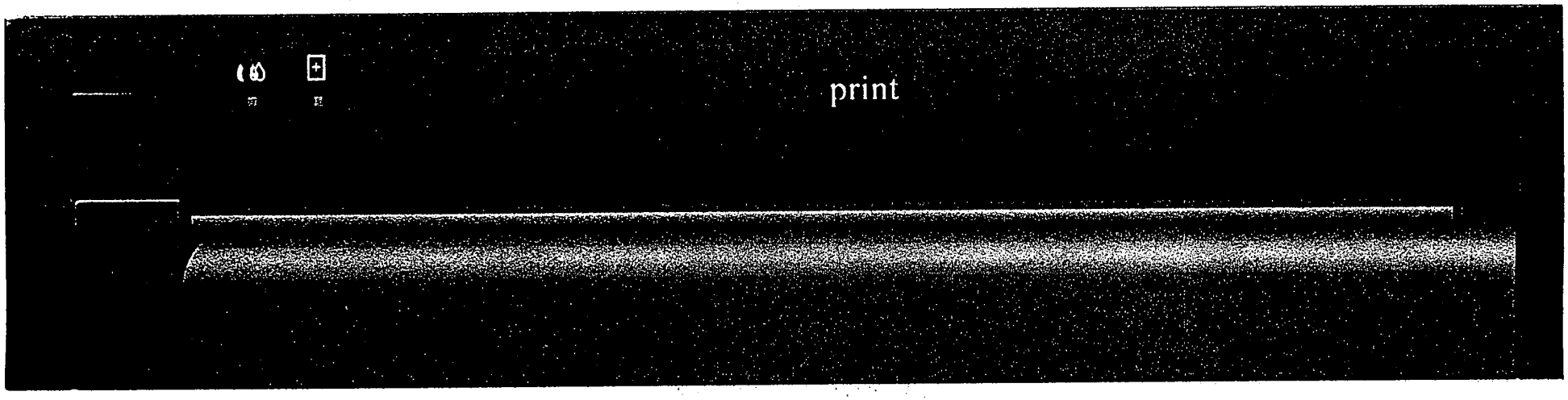


Figure 4. Front panel of double-sided version of CePrint OEM printer shown actual size (60mm h x 272mm w)

3 Memjet-Based Printing

A Memjet printhead produces 1600 dpi bi-level CMYK. On low-diffusion paper, each ejected drop forms an almost perfectly circular 22.5μm diameter dot. Dots are easily produced in isolation, allowing dispersed-dot dithering to be exploited to its fullest. Since the Memjet printhead is page-width and operates with a constant paper velocity, the four color planes are printed in perfect registration, allowing ideal dot-on-dot printing. Since there is consequently no spatial interaction between color planes, the same dither matrix is used for each color plane.

A page layout may contain a mixture of images, graphics and text. Continuous-tone (contone) images and graphics are reproduced using a stochastic dispersed-dot dither. Unlike a clustered-dot (or amplitude-modulated) dither, a *dispersed-dot* (or frequency-modulated) dither reproduces high spatial frequencies (i.e. image detail) almost to the limits of the dot resolution, while simultaneously reproducing lower spatial frequencies to their full color depth, when spatially integrated by the eye. A *stochastic* dither matrix is carefully designed to be free of objectionable low-frequency patterns when tiled across the image. As such its size typically exceeds the minimum size required to support a number of intensity levels (i.e. 16×16×8 bits for 257 intensity levels). CePrint uses a dither *volume* of size 64×64×3×8 bits. The volume provides an extra degree of freedom during the design of the dither by allowing a dot to change states multiple times through the intensity range (rather than just once as in a conventional dither matrix).

Human contrast sensitivity peaks at a spatial frequency of about 3 cycles per degree of visual field and then falls off logarithmically, decreasing by a factor of 100 beyond about 40 cycles per degree and becoming immeasurable beyond 60 cycles per degree [2,4]. At a normal viewing distance of 12 inches (about 300mm), this translates roughly to 200-300 cycles per inch (cpi) on the printed page, or 400-600 samples per inch according to Nyquist's theorem. Contone resolution beyond about 400 pixels per inch (ppi) is therefore of limited utility, and in fact contributes slightly to color error through the dither.

Black text and graphics are reproduced directly using bi-level black dots, and are therefore not antialiased (i.e. low-pass filtered) before being printed. Text is therefore *supersampled* beyond the perceptual limits discussed above, to produce smoother edges when spatially integrated by the eye. Text resolution up to about 1200 dpi continues to contribute to perceived text sharpness (assuming low-diffusion paper, of course).

4 Page Delivery Architecture

4.1 PAGE IMAGE SIZES

CePrint prints A4 and Letter pages with full edge bleed. Corresponding page image sizes are set out in Table 2 for various spatial resolutions and color depths used in the following discussion. Note that the size of an A4 page exceeds the size of a Letter page, although the Letter page is wider. Page buffer requirements are therefore based on A4, while line buffer requirements are based on Letter.

Table 2. Page Image Sizes

spatial resolution (dots/inch)	color depth (bits/pixel)	A4[a] page buffer size	Letter[b] page buffer size
1600	32	948MB	913MB
800	32	237MB	228MB
400	32	59.3MB	57.1MB
267	32	26.4MB	25.4MB
1600	4	119MB	114MB
800	4	29.6MB	28.5MB
1600	1	29.6MB	28.5MB
800	1	7.4MB	7.1MB

a. 210mm × 297mm, or 8.3" × 11.7"
b. 8.5" × 11"

4.2 CONSTRAINTS

The act of interrupting a Memjet-based printer during the printing of a page produces a visible discontinuity, so it is advantageous for the printer to receive the entire page before commencing printing, to eliminate the possibility of buffer underrun. Furthermore, if the transmission of the page from the host to the printer takes significant time in relation to the time it takes to print the page, then it is advantageous to provide two page buffers in the printer so that one page can be printed while the next is being received. If the transmission time of a page is less than its 2-second printing time, then double-buffering allows the full 30 pages/minute page rate of CePrint to be achieved.

Figure 5 illustrates the sustained printing rate achievable with double-buffering in the printer, assuming 2-second page rendering and 2-second page transmission.

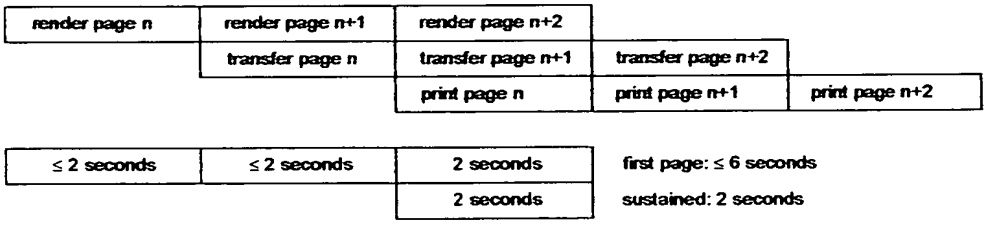


Figure 5. First page and sustained printing rate

Assuming it is economic for the printer to have a maximum of only 8MB of memory (i.e. a single 64Mbit DRAM), then less than 4MB is available for each page buffer in the printer, imposing a limit of less than 4MB on the size of the page image. To allow for program and working memory in the printer, we limit this to 3MB per page image.

Assuming the printer has only a typical low-speed connection to the host processor, then the speed of this connection is 1-2MB/s (i.e. 2MB/s for parallel port, 1.5MB/s for USB, and 1MB/s for 10Base-T Ethernet). Assuming 2-second page transmission (i.e. equal to the printing time), this imposes a limit of 2-4MB on the size of the page image, i.e. a limit similar to that imposed by the size of the page buffer.

In practice, because the host processor and the printer can be closely coupled, a high-speed connection between them may be feasible.

Whatever the speed of the host connection required by the single-sided version of CePrint, the double-sided version requires a connection of twice that speed.

4.3 PAGE RENDERING AND COMPRESSION

Page rendering (or rasterization) can be split between the host processor and printer in various ways. Some printers support a full page description language (PDL) such as Postscript, and contain correspondingly sophisticated renderers. Other printers provide special support only for rendering text, to achieve high text resolution. This usually includes support for built-in or downloadable fonts. In each case the use of an embedded renderer reduces the rendering burden on the host processor and reduces the amount of data transmitted from the host processor to the printer. However, this comes at a price. These printers are more complex than they might be, and are often unable to provide full support for the graphics system of the host, through which application programs construct, render and print pages. They fail to exploit the possibly high performance of the host processor.

CePrint relies on the host processor to render pages, i.e. contone images and graphics to the pixel level, and black text and graphics to the dot level. CePrint contains only a simple rendering engine which dithers the contone data and combines the results with any foreground bi-level black text and graphics. This strategy keeps the printer simple, and independent of any page description language or graphics system. It fully exploits the high performance expected in the host processor of a multimedia CE device. The downside of this strategy is the potentially large amount of data which must be transmitted from the host processor to the printer. We therefore use compression to reduce the page image size to the 3MB required to allow a sustained printing rate of 30 pages/minute.

An 8.3" × 11.7" A4 page has a bi-level CMYK page image size of 119MBytes at 1600 dpi, and a contone CMYK pagesize of 59.3MB at 400 ppi.

We use JPEG compression to compress the contone data. Although JPEG is inherently lossy, for compression ratios of 10:1 or less the loss is usually negligible [17]. To achieve a high-quality compression ratio of less than 10:1, and to obtain an integral contone to bi-level ratio, we choose a contone resolution of 267 ppi. This yields a contone CMYK pagesize of 25.5MB, a corresponding compression ratio of 8.5:1 to fit within the 3MB/page limit, and a contone to bi-level ratio of 1:6 in each dimension.

A full page of black text (and/or graphics) rasterized at printer resolution (1600 dpi) yields a bi-level image of 29.6MB. Since rasterizing text at 1600 dpi places a heavy burden on the host processor for a small gain in quality, we choose to rasterize text at 800 dpi. This

yields a bi-level image of 7.4MB, requiring a lossless compression ratio of less than 2.5:1 to fit within the 3MB/page limit. We achieve this using a two-dimensional bi-level compression scheme similar to the compression scheme used in Group 4 Facsimile.

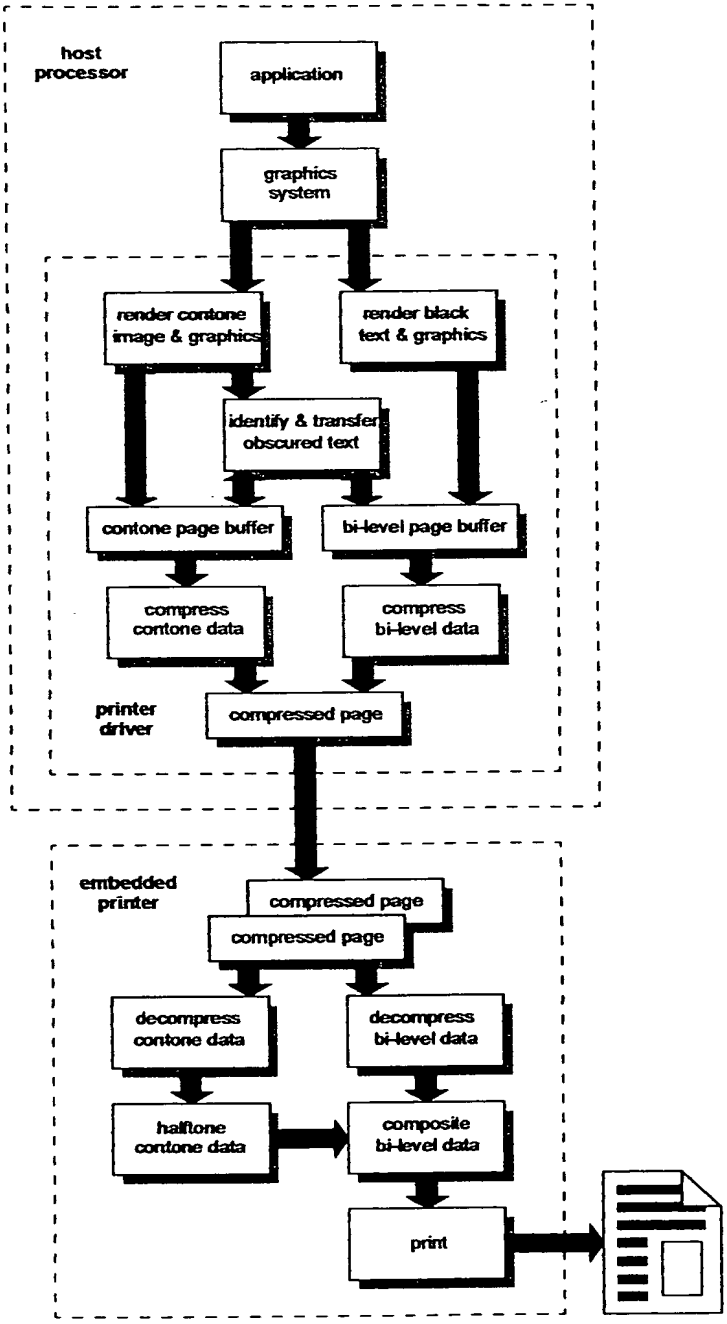


Figure 6. Conceptual data flow from application to printed page

As long as the image and text regions of a page are non-overlapping, any combination of the two fits within the 3MB limit. If text lies on top of a background image, then the worst case is a compressed page image size approaching 6MB (depending on the actual text compression ratio). This fits within the printer's page buffer memory, but prevents double-buffering of pages in the printer, thereby reducing the printer's page rate by two-thirds, i.e. to 10 pages/minute.

Figure 7 on the next page shows a typical page containing images and text which compresses to about 1.5MB.

4.4 PAGE EXPANSION AND PRINTING

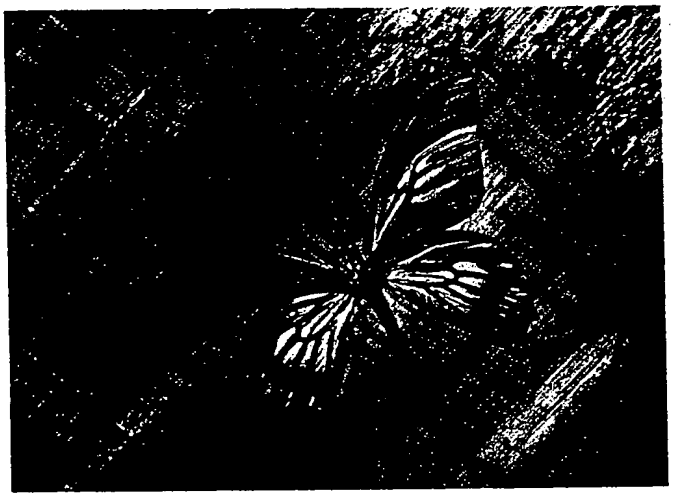
As described above, the host processor renders contone images and graphics to the pixel level, and black text and graphics to the dot level. These are compressed by different means and transmitted together to the printer.

The printer contains two 3MB page buffers - one for the page being received from the host, and one for the page being printed. The printer expands the compressed page as it is being printed. This expansion consists of decompressing the 267 ppi contone CMYK image data, halftoning the resulting contone pixels to 1600 dpi bi-level CMYK dots, decompressing the 800 dpi bi-level black text data, and compositing the resulting bi-level black text dots *over* the corresponding bi-level CMYK image dots.

The conceptual data flow from the application to the printed page is illustrated in Figure 6.

Figure 7. A sample A4 page (reduced) which compresses to about 1.5MB

When we look to the individuals of the same variety or sub-variety of our older cultivated plants and animals, one of the first points which strikes us, is, that they generally differ much more from each other, than do the individuals of any one species or variety in a state of nature. When we reflect on the vast diversity of the plants and animals which have been cultivated, and which have varied during all ages under the most different climates and treatment, I think we are driven to conclude that this greater variability is simply due to our domestic productions having been raised under conditions of life not so uniform as, and somewhat different from, those to which the parent-species have been exposed under nature. There is, also, I think, some probability in the view propounded by Andrew Knight, that this variability may be partly connected with excess of food. It seems pretty clear that organic beings must be exposed during several generations to the new conditions of life to cause any appreciable amount of variation; and that when the organisation has once begun to vary, it generally continues to vary for many generations. No case is on record of a variable being ceasing to be variable under cultivation. Our oldest cultivated plants, such as wheat, still often yield new varieties: our oldest domesticated animals are still capable of rapid improvement or modification. It has been disputed at what period of life the causes of variability, whatever they may be, generally act; whether during the early or late period of development of the embryo, or at the instant of conception. Geoffroy St. Hilaire's experiments show that unnatural treatment of the embryo causes monstrosities; and monstrosities cannot be separated by any clear line of distinction from mere variations. But I am strongly inclined to suspect that the most frequent cause of variability may be attributed to the male and female reproductive elements having been affected prior to the act of conception. Several reasons make me believe in this; but the chief one is the remarkable effect which confinement or cultivation has on the functions of the reproductive system; this system appearing to be far more susceptible than any other part of the organisation, to the action of any change in the conditions of life. Nothing is more easy than to tame an animal, and few things more difficult than to get it to breed freely under confinement, even in the many cases when the male and female unite. How many animals there are which will not breed, though living long under not very close confinement in their native country! This is generally attributed to vitiated instincts; but how many cultivated plants display the utmost vigour, and yet rarely or never seed! In some few such cases it has been found out that very trifling changes, such as a little more or less water at some particular period of growth, will determine whether or not the plant sets a seed. I cannot here enter on the copious details which I have collected on this curious subject; but to show how singular the laws are which determine the reproduction of animals under confinement, I may just mention that carnivorous animals, even from the tropics, breed in this country pretty freely under confinement, with the exception of the plantigrades or bear family; whereas, carnivorous birds, with the rarest exceptions, hardly ever lay fertile eggs. Many exotic plants have pollen utterly worthless, in the same exact condition as in the most sterile hybrids. When, on the one hand, we see domesticated animals and plants, though often



5 Printer Hardware

CePrint is conceived as an OEM part designed for inclusion primarily in consumer electronics (CE) devices. Intended markets include televisions, VCRs, PhotoCD players, DVD players, Hi-fi systems, Web/Internet terminals, computer monitors, and vehicle consoles. It features a low-profile front panel and provides user access to paper and ink via an ejecting tray. It operates in a horizontal orientation under domestic environmental conditions.

Because of the simplicity of the pagewidth Memjet printhead, CePrint contains an ultra-compact print mechanism which yields an overall product height of 40mm for the single-sided version and 60mm for the double-sided version.

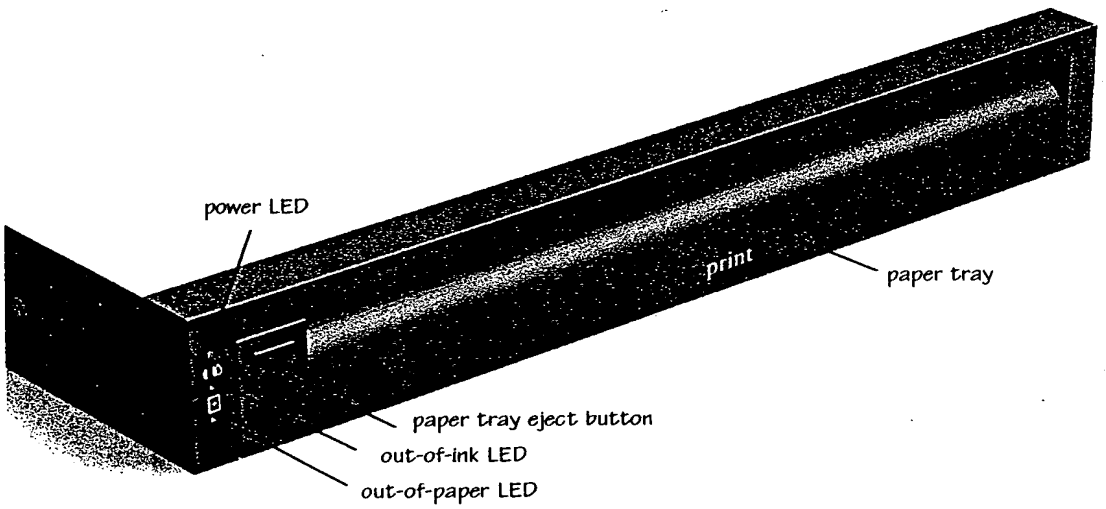


Figure 8. Front panel of the single-sided CePrint OEM printer unit (40mm h × 272mm w)

The nature of an OEM product dictates that it should be simple in style and reflect minimum dimensions for inclusion into host products. CePrint is styled to be accommodated into all of the target market products and has minimum overall dimensions of 40mm high × 272mm wide × 416 mm deep. The only cosmetic part of the product is the front fascia and front tray plastics. These can be re-styled by a manufacturer if they wish to blend CePrint with a certain piece of equipment.

CePrint is basically a motorized A4/Letter paper tray with a removable ink cartridge and a Memjet printhead mechanism. It comprises a metal chassis, fascias, a metal cover, a molded paper tray, ink cartridge, 3 motors, a flex PCB, a rigid PCB and various injection moldings and smaller parts to achieve a cost effective high volume product.

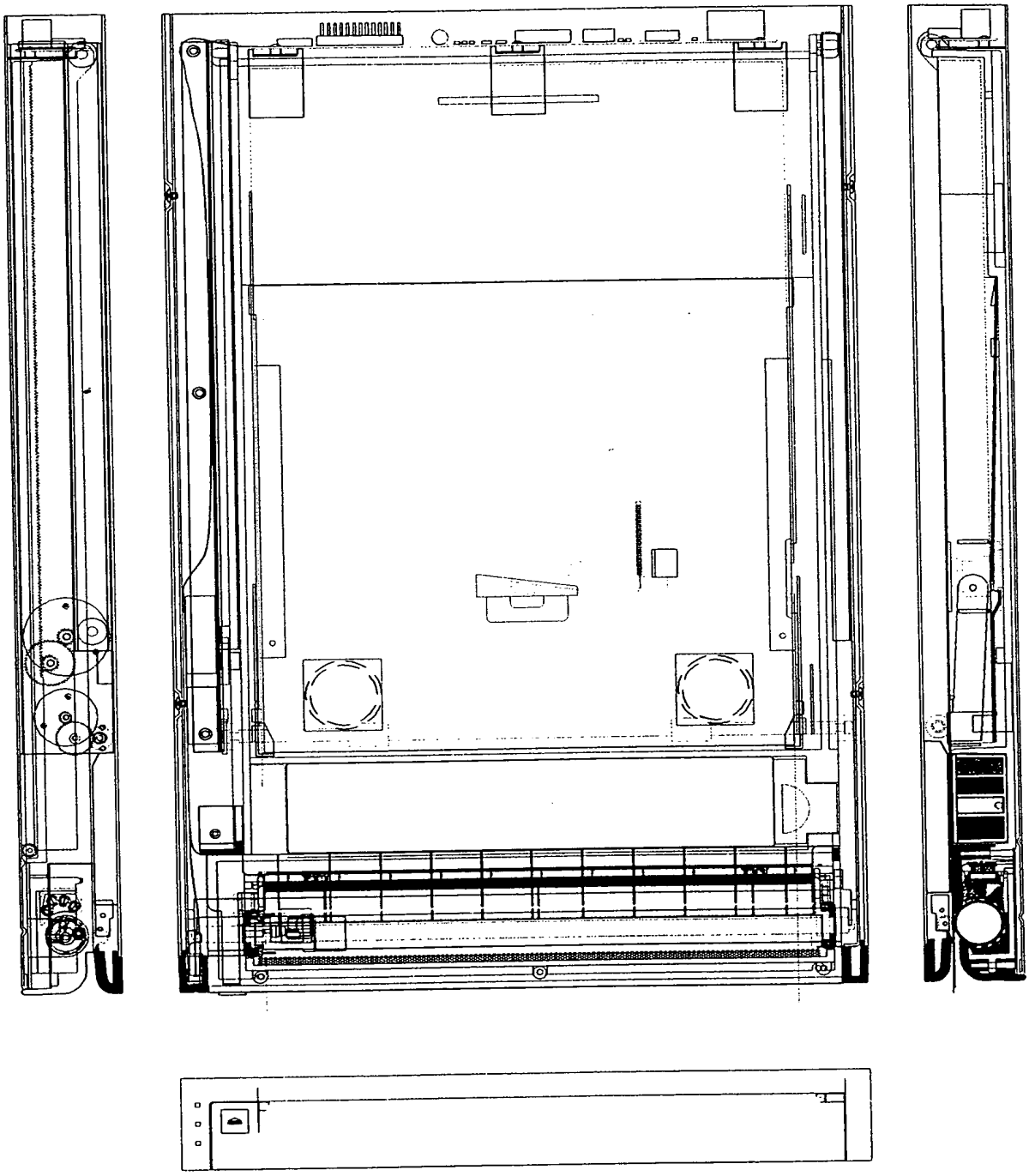


Figure 9. Plan and elevations of CePrint OEM printer unit

The product is simple in operation, only requiring user interaction when paper or ink need replenishing as indicated by the front-panel LEDs. The paper handling mechanisms are similar to current printer applications and are therefore considered reliable. In the rare event of a paper jam, the action of ejecting the paper tray allows the user to deal with the problem. The tray has a sensor which retracts the tray if the tray is pushed. If the tray jams on the way in, this is also sensed and the tray is re-ejected. This allows the user to be lazy in operating the tray by just pushing it to close and protects the unit from damage should the tray get knocked while in the out position. It also caters for children sticking fingers in the tray while closing. Ink is replaced by inserting a new cartridge into the paper tray and securing it by a cam lock lever mechanism.

5.1 OVERVIEW

The overall views of CePrint are shown in Figure 10 and Figure 11. Figure 9 gives the context of these views. The elevation in Figure 11 shows the base metalwork (A3), base tray roller wheels (A4) and a bracket (A5) that accommodates the motors (M2 & M5) and gears (M1 & M3) for ejecting the paper tray and driving the paper pick-up roller.

The plan in Figure 10 shows the major features of the product. Attached to the bracket (A5) and the base metalwork (A3) are two guide rails (A7) that allow the molded paper tray (D1) and its roller wheels (D2) to slide forward. The tray sits on rollers (A4) towards the front of the base metalwork and this provides a strong, low friction and steady method of ejection and retraction. A flex PCB (E3) runs from the main PCB (E7) via the motors to a contact molding (E1) and the lightpipe area (C2). An optical sensor on the flex PCB allows the tray ejector motor (M5) to retract the tray independently of the eject button (C3) if the tray is pushed when in the out position by sensing a hole in the gear wheel (M4). Similarly, the tray is ejected if there is any stoppage during retraction.

The contact molding (E1) has a foam pad (E2) that the flex PCB is fixed onto and provides data and power contacts to the printhead and bus bars during printing. The transfer roller (P6 in Figure 12) has two end caps (P19) that run in low friction bearing assemblies (P20). One of the end caps has an internal gear that acts on a small gear (M10), which transfers power through a reduction gear (M9) to a worm drive. This is reduced further via another gear to the motor worm drive (M7). This solution for a motor drive assembly that is housed inside the transfer roller saves space for future designs and mounts onto a small chassis (M11) that is attached to the ink connector moldings (B1 & B2).

The ink connector has four pins (B3) with an ejector plate (B4) and springs (B5) that interfaces with the ink cartridge. The ink cartridge is accessed via a cam lock lever and spring (D11). The ink is conducted through molded channels into a flexible four-channel hose connector (B6) that interfaces with the printhead cartridge end cap (P22). The other end of the printhead cartridge has a different flexible sealing connector (P21) on the end cap to allow ink to be drawn through the cartridge during assembly and effectively charge the unit and ink connector with ink in a sealed environment. The printhead and ink connector assemblies are mounted directly into the paper tray (D1).

The paper tray has several standard paper handling components, namely a metal base channel (D3) with low friction pads (D8), sprung by two compression springs (D9) and two metal paper guides (D6) with arms (D10) secured by rivets. The paper is aligned to one side of the tray by a spring steel clip (D7). The tray is normally configured to take A4 paper, but Letter size paper is accommodated by relocating one of the paper guide assemblies and clipping a plate into the paper tray to provide a rear stop. When empty the paper tray can be filled with up to 150 sheets.

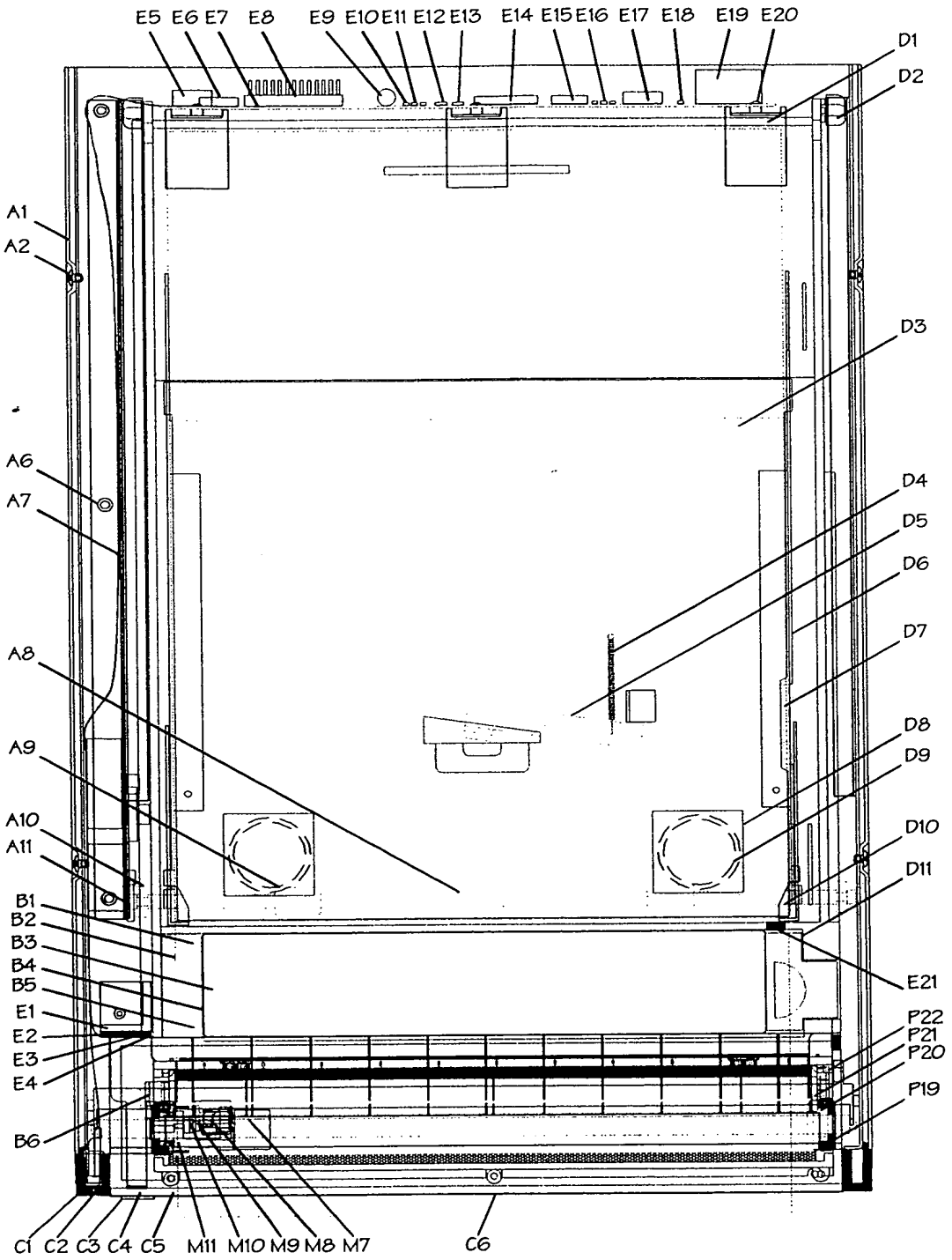


Figure 10. Plan of CePrint OEM printer unit, with part references

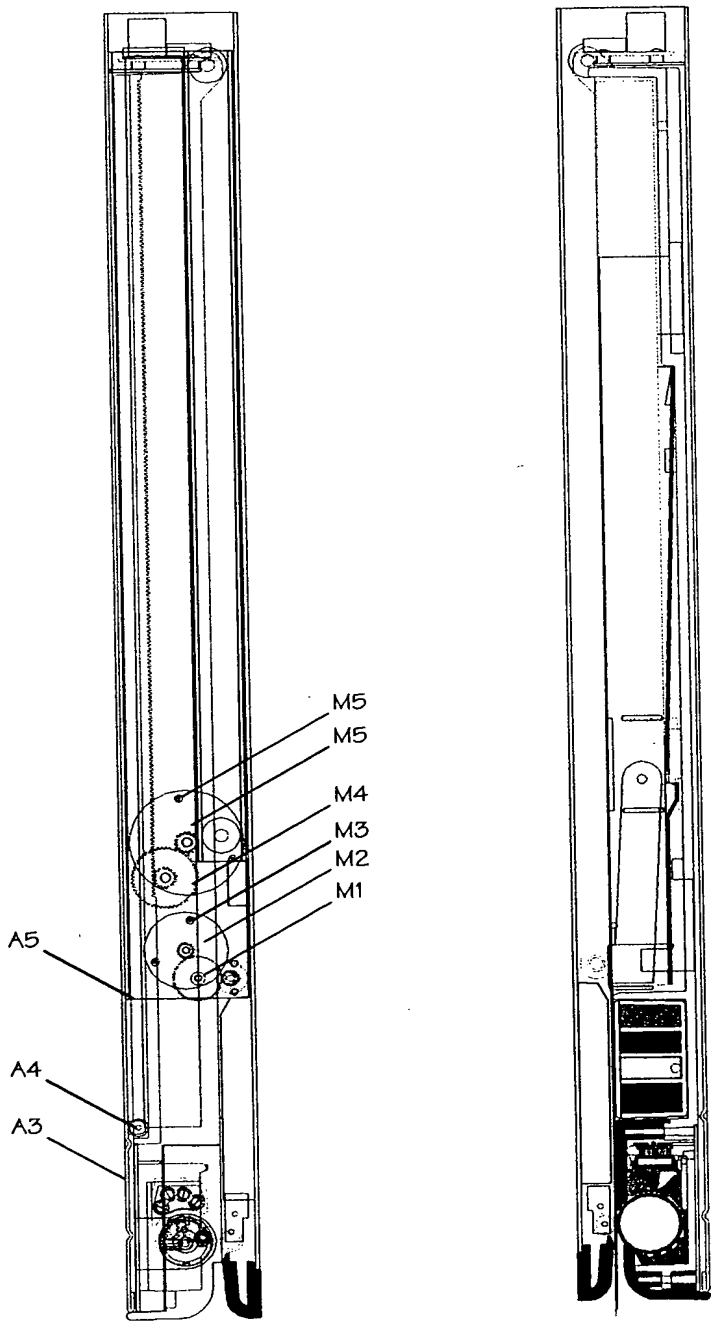


Figure 11. Left and right side elevations of CePrint OEM printer unit, with part references

As is standard practice for adding paper, the metal base channel (D3) is pushed down and is latched using the tray lock molding (D5) and the return spring (D4). When paper has been added and the tray retracted, the tray lock molding is unlatched by hitting a metal return in the base metalwork.

The unit is now ready to print. When activated, the paper pick-up roller (A8 & A9) is driven by a small drive gear (A10) that meshes with another drive gear (M1) and a normal motor (M5). The roller is located to the base metalwork by two heat staked retainer moldings (A11). A small molding on the end of the pick-up roller acts with a sensor on the flex PCB to accurately position the pick-up roller in a parked position so that paper and tray can be withdrawn without touching it when ejecting. This accurate positioning also allows the roller to feed the sheet to the transfer roller with a fixed number of revolutions. As the transfer roller is running at a similar speed there should be no problem with take-up of the paper. An optical sensor (E21) mounted into a the cover molding (C5) finds the start of each sheet and engages the transfer motor, so there is no problem if for example a sheet has moved forward of the roller during any strange operations.

The main PCB (E7) is mounted onto the base metalwork via standard PCB standoffs (E20) and is fitted with a data connector (E8) and a DC connector (E19). The front fascia (C1) is mounted onto the base metalwork using snap details and a top metal cover (A1) completes the overall product with RFI/EMI integrity via four fixings (A2).

5.2 PRINTHEAD ASSEMBLY AND IMAGE TRANSFER MECHANISM

The Memjet printhead assembly is shown in green in Figure 12. This represents one of the four possible ways to deploy the printhead in conjunction with the ink cartridge in a product such as CePrint:

- permanent printhead, replaceable ink cartridge (as shown here)
- separate replaceable printhead cartridge and ink cartridge
- refillable combined printhead and ink cartridge
- disposable combined printhead and ink cartridge

The Memjet printhead (P9) prints onto the titanium nitride (TiN) coated transfer roller (P6) which rotates in an anticlockwise direction to transfer the image onto the paper (shown as a red horizontal line). The paper is pressed against the transfer roller by the spring-loaded rubber coated pinch roller (P17). After transferring the image to the paper the transfer roller continues past the cleaning sponge (P4) and finally the rubber wiper (rightmost part of P5).

While operational, the printhead is held off the transfer roller by a solenoid (P13). When not operational, the printhead is parked against the transfer roller. The printhead's integral elastomeric seal (P10) seals the printhead and prevents it from drying out. The two printhead positions are illustrated in Figure 13.

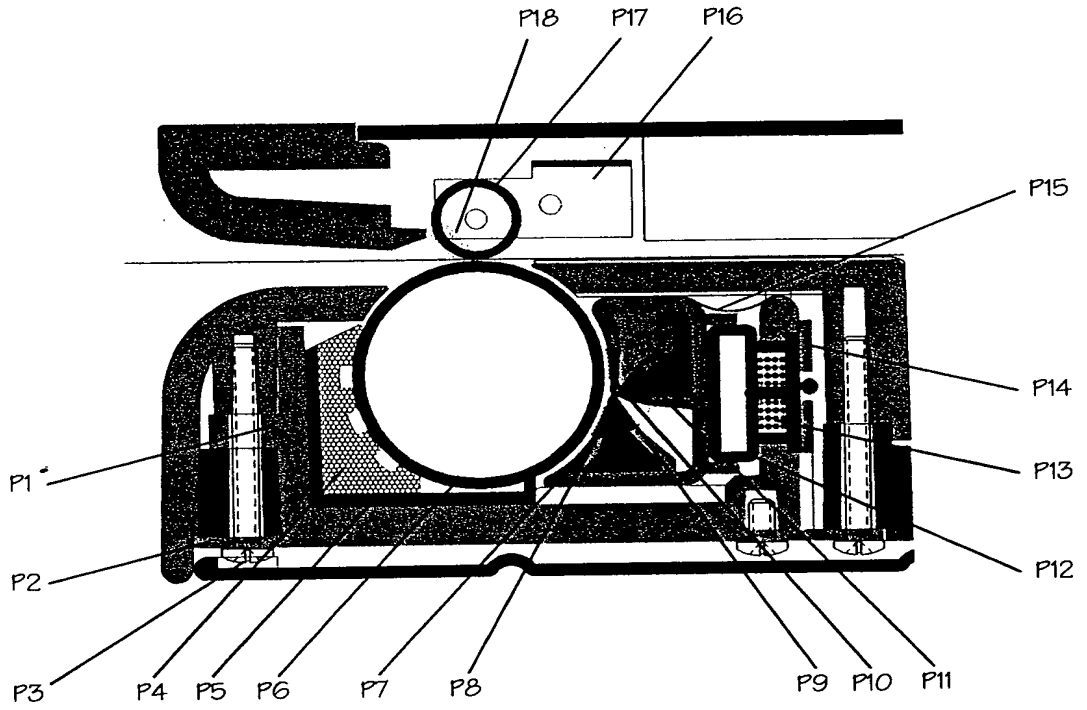


Figure 12. Cross-section of printhead assembly and image transfer mechanism, with part references

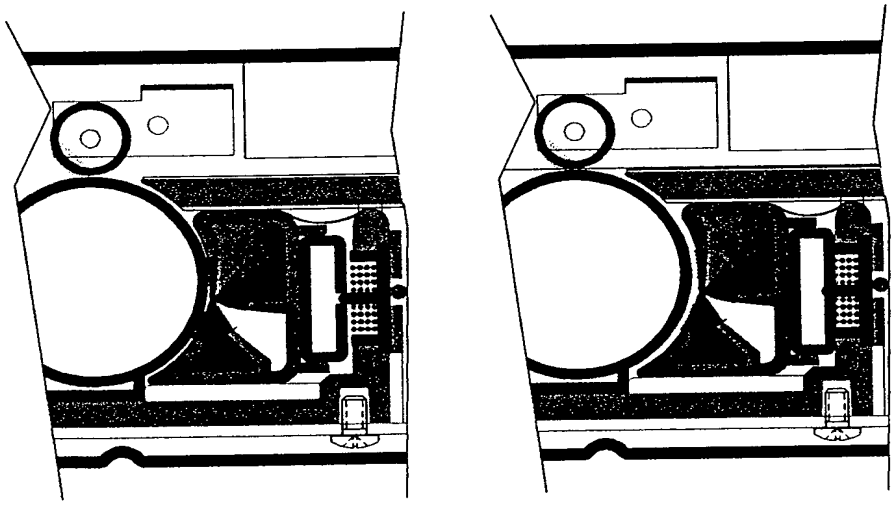


Figure 13. Printhead parked and sealed against the transfer roller (left), and in operation (right)

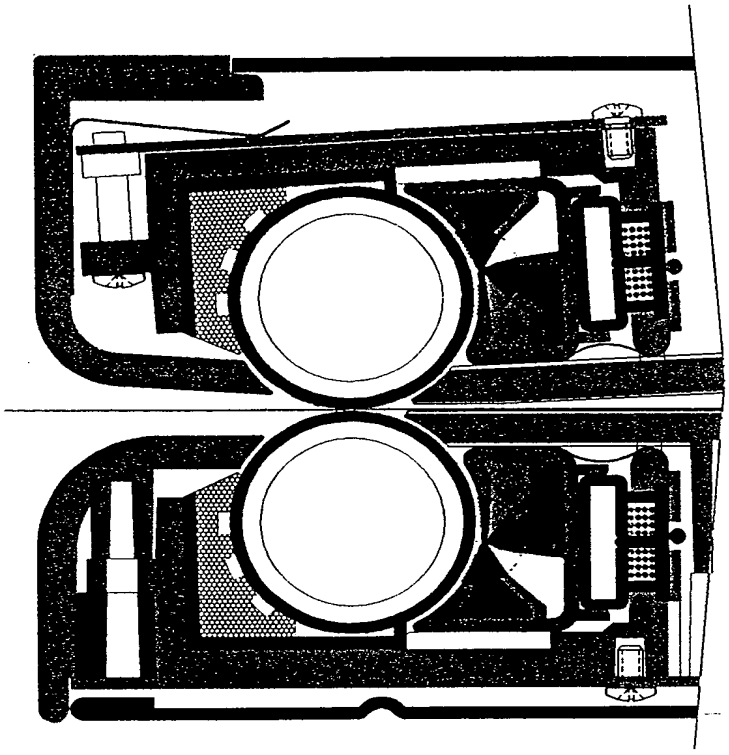


Figure 14. Dual printheads and transfer rollers mounted in opposition and providing double-sided printing

In the double-sided version of CePrint, there are dual printheads and transfer rollers, mounted in opposition as illustrated in Figure 14. The lower assembly is fixed while the upper assembly pivots and is sprung to press against the paper. The upper transfer roller takes the place of the pinch roller in the single-sided version

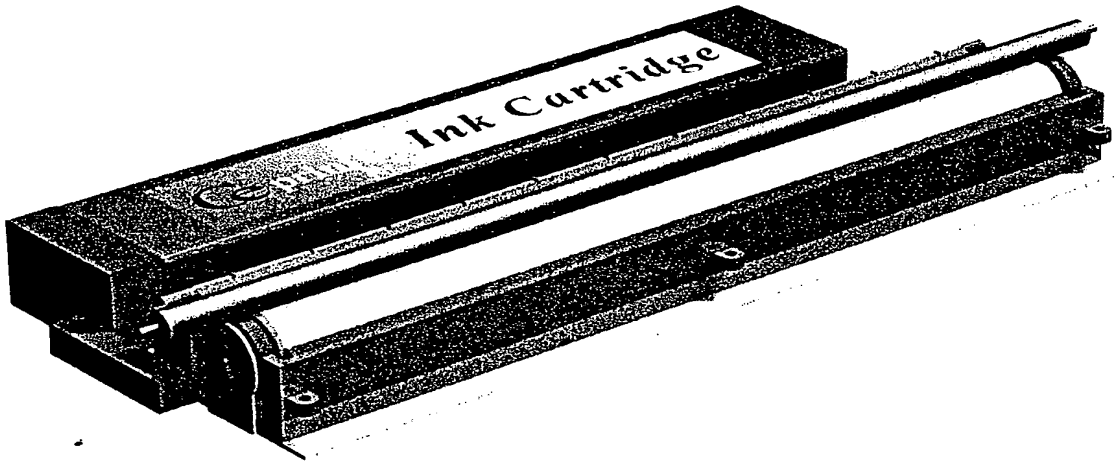


Figure 15. Memjet CePrint printer engine

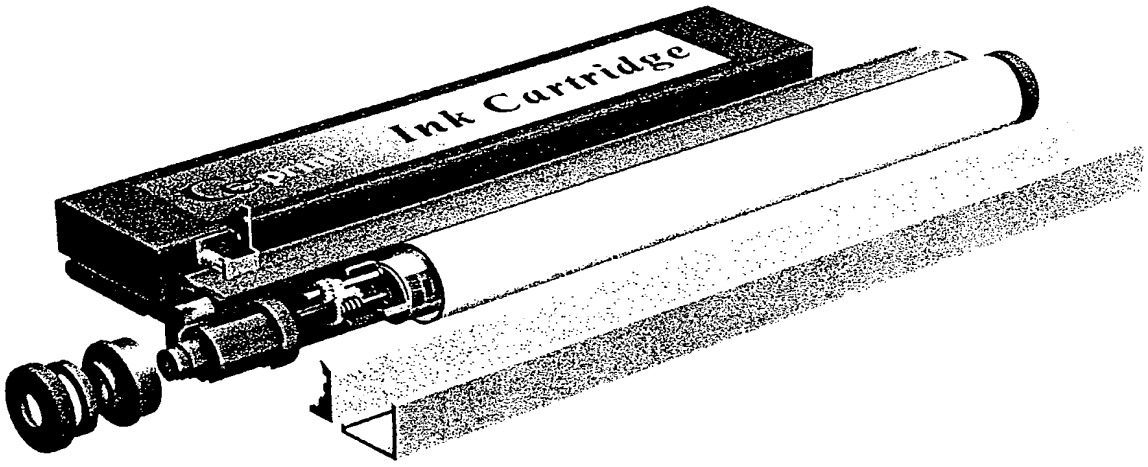


Figure 16. Exploded view of paper drive chain

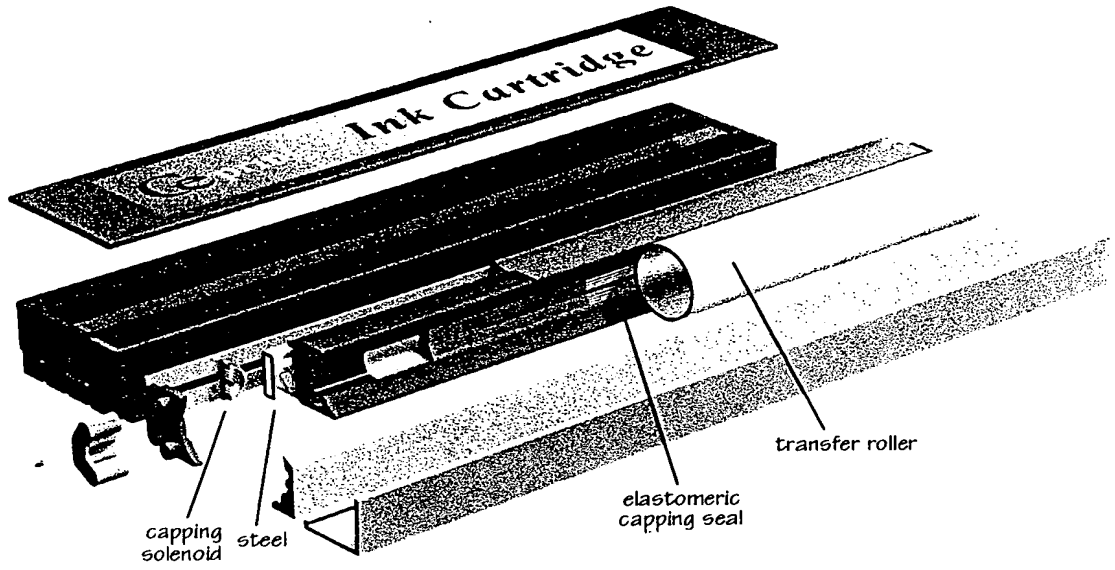


Figure 17. Cut-away view of printing system

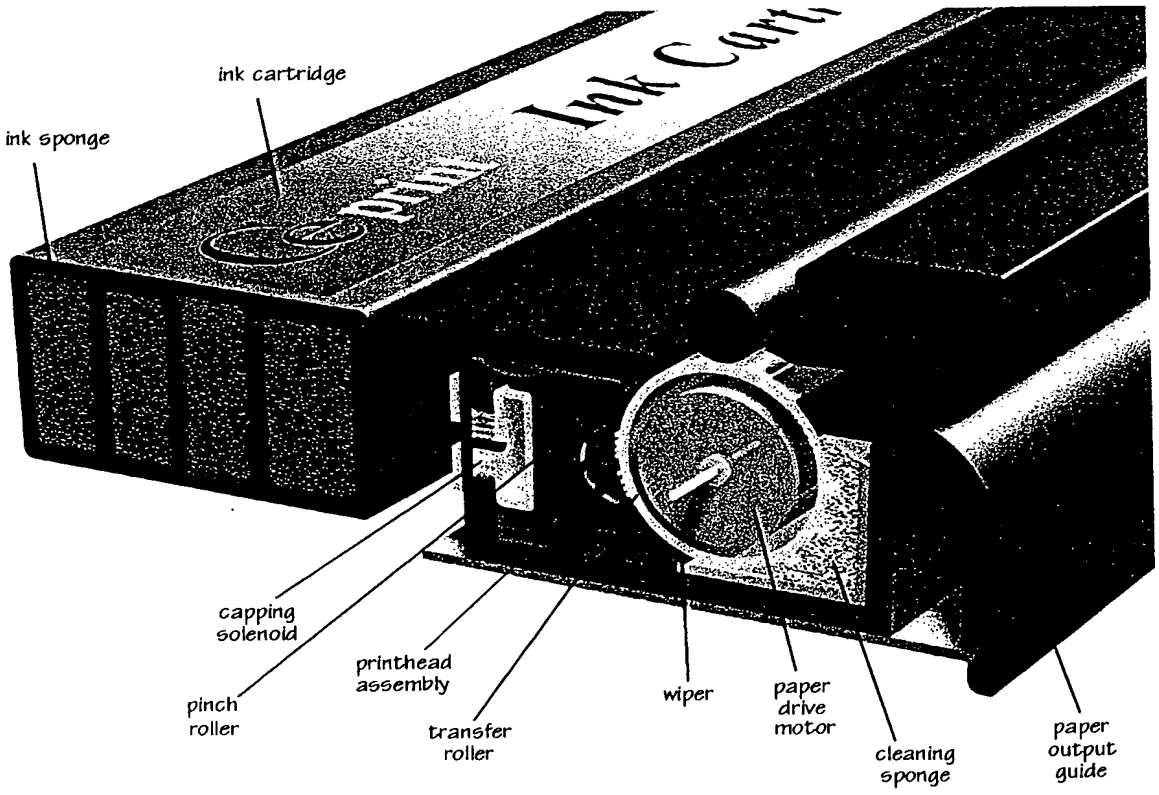


Figure 18. Detail of printing system

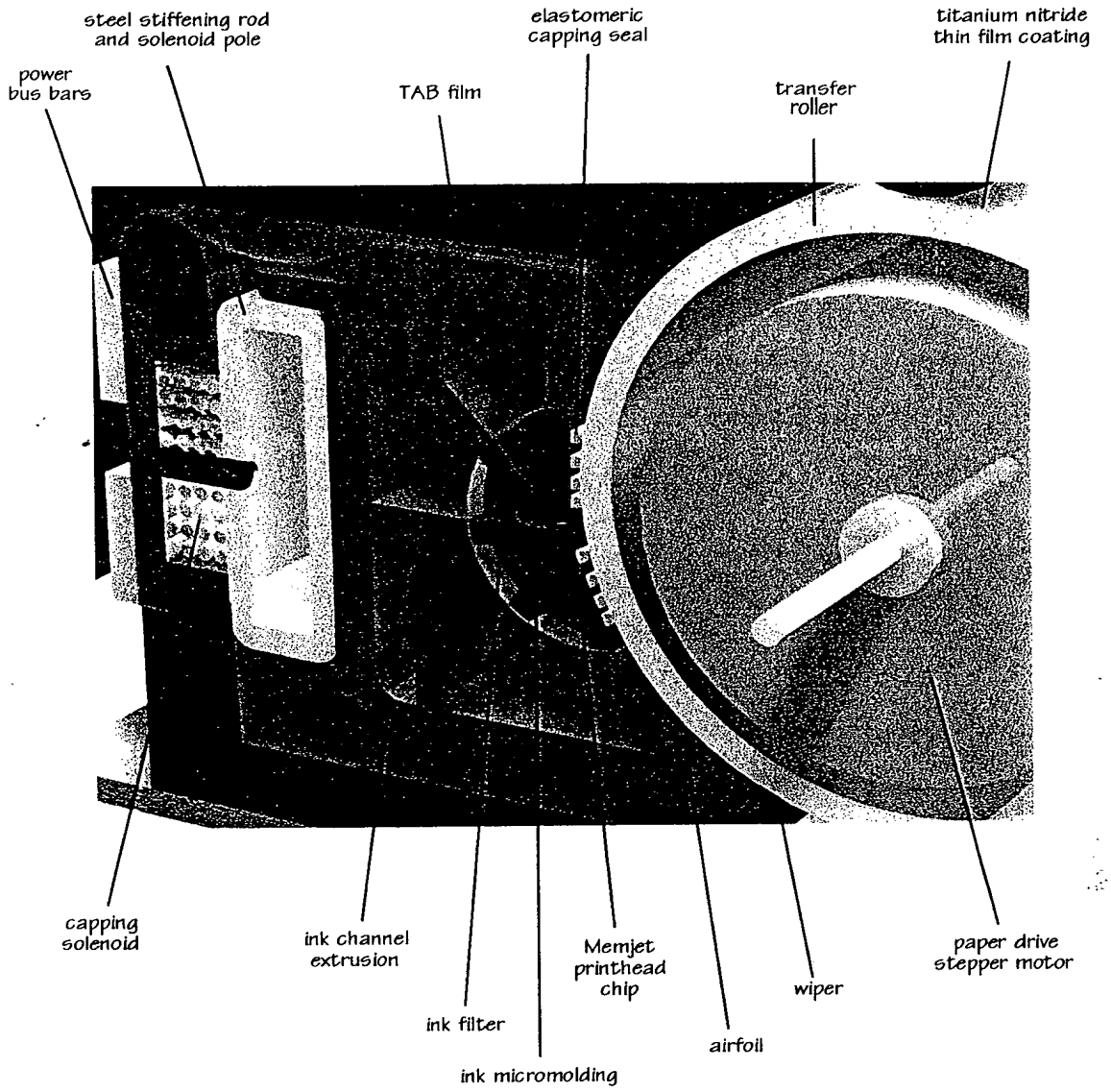


Figure 19. Fine detail of printing system

5.3 MANUFACTURING COST

The manufacturing cost for CePrint is given in Table 3, assuming at least 10 million units per annum in 2001. The manufacturing cost for the ink cartridge is given in Table 4.

Part references refer to Figure 10 and Figure 11 (plan and elevations of CePrint unit - part prefixes A through E), and Figure 12 (cross-section of printhead and image transfer mechanism - part prefix P).

Table 3. CePrint manufacturing cost

part ref.	item	quantity	unit cost	cost
A1	Top cover metalwork	1	$0.200	$0.20
A2	Top cover screws	4	$0.005	$0.02
A3	Base metalwork	1	$0.300	$0.30
A4	Base metalwork roller wheels and pins	2	$0.010	$0.02
A5	Bracket	1	$0.100	$0.10
A6	Bracket screws	3	$0.005	$0.02
A7	Guide rails	2	$0.100	$0.20
A8	Paper feed s/steel rod	1	$0.020	$0.02
A9	Paper feed rod grippers	2	$0.010	$0.02
A10	Paper feed gear	1	$0.010	$0.01
A11	Paper feed retainer moldings	2	$0.010	$0.02
B1	Connector molding front	1	$0.050	$0.05
B2	Connector molding rear	1	$0.050	$0.05
B3	Connector molding pins	4	$0.010	$0.04
B4	Connector molding ejector plate	1	$0.010	$0.01
B5	Springs	2	$0.010	$0.02
B6	Flexible Ink channel molding	1	$0.010	$0.01
C1	Front facia molding	1	$0.100	$0.10
C2	Lightpipe molding	1	$0.010	$0.01
C3	Button molding	1	$0.010	$0.01
C4	Spring	1	$0.005	$0.01
C5	Cover molding	1	$0.700	$0.70
C6	Printed logo	1	$0.005	$0.01
D1	Paper tray molding	1	$0.200	$0.20
D2	Paper tray nylon roller wheels	2	$0.005	$0.01
D3	Metal base channel	1	$0.020	$0.02
D4	Tray lock spring	1	$0.005	$0.01
D5	Tray lock molding	1	$0.010	$0.01
D6	Paper guides (metal)	2	$0.020	$0.04
D7	Paper aligning spring clip	1	$0.010	$0.01
D8	Pads	2	$0.005	$0.01
D9	Compression springs	2	$0.010	$0.02

Table 3. CePrint manufacturing cost

part ref.	item	quantity	unit cost	cost
D10	Paper guide arms (metal) & rivet	2	$0.010	$0.02
D11	Ink cartridge cam lock molding & spring	1	$0.030	$0.03
P1	Printhead base molding	1	$0.030	$0.03
P2	Printhead base metalwork	1	$0.010	$0.01
P3	Screws	5	$0.005	$0.03
P4	Cleaning sponge	1	$0.005	$0.01
P5	Cleaning co-extrusion with rubber wiper	1	$0.010	$0.01
P6	Transfer roller	1	$0.200	$0.20
P7	Printhead cartridge extrusion	1	$0.020	$0.02
P8	Printhead cartridge precision molding	1	$0.100	$0.10
P9	Printhead chip	2	$3.000	$6.00
P10	Printed elastomeric seal	1	$0.020	$0.02
P11	Printhead cartridge filter	1	$0.100	$0.10
P12	Stainless steel extruded keeper (adhesive backed)	1	$0.020	$0.02
P13	Solenoid	2	$0.100	$0.20
P14	Copper bus bars (gold plated)	2	$0.100	$0.20
P15	Printhead cartridge TAB film	1	$1.000	$1.00
P16	Pinch roller cradle metalwork & pins	1	$0.020	$0.02
P17	Pinch roller rubber coated	1	$0.020	$0.02
P18	Pinch roller cradle spring	1	$0.005	$0.01
P19	Transfer roller end moldings	2	$0.010	$0.02
P20	Ball bearing races	2	$0.020	$0.04
P21	Printhead cartridge ink exit rubber molding	1	$0.010	$0.01
P22	Printhead cartridge end caps	2	$0.010	$0.02
E1	Contact molding	1	$0.010	$0.01
E2	Foam contact pad	1	$0.005	$0.01
E3	Flex PCB & bus bar contacts	1	$0.100	$0.10
E4	Flex PCB optical sensor for paper tray	1	$0.100	$0.10
E5	Flex PCB connector	1	$0.050	$0.05
E6	Motor connector	1	$0.050	$0.05
E7	Rigid PCB (double sided)	1	$1.000	$1.00
E8	Data connector	1	$0.100	$0.10
E9	Ceramic decoupling capacitors	4	$0.005	$0.02
E10	Tantalum decoupling capacitors	4	$0.010	$0.04
E11	Electrolytic capacitors	4	$0.020	$0.08
E12	Motor drive transistors	4	$0.010	$0.04
E13	Voltage regulator	1	$0.100	$0.10
E14	CePrint central processor (CCP) ASIC (0.18 micron)	1	$4.000	$4.00
E15	64Mbit RDRAM (year 2001, 0.15 micron)	1	$4.000	$4.00

Table 3. CePrint manufacturing cost

part ref.	item	quantity	unit cost	cost
E16	Surge protector	1	$0.020	$0.02
E17	QA chip	1	$0.100	$0.10
E18	Surface mount resistors	14	$0.005	$0.07
E19	DC connector	1	$0.050	$0.05
E20	PCB stand off moldings (standard)	6	$0.005	$0.03
E21	Optomechanical Media Sensor	1	$0.200	$0.20
M1	Drive gear & pin	1	$0.010	$0.01
M2	Paper tray ejector motor	1	$0.300	$0.30
M3	Screws	2	$0.005	$0.01
M4	Drive gear & pin	1	$0.010	$0.01
M5	Paper pick-up motor	1	$0.200	$0.20
M6	Screws	2	$0.005	$0.01
M7	Transfer roller stepper motor, metal plate and worm gear	1	$0.500	$0.50
M8	Bracket	1	$0.010	$0.01
M9	Reduction gear (nylon)	1	$0.005	$0.01
M10	Worm gear, pin and small drive gear	1	$0.020	$0.02
M11	Motor chassis molding	1	$0.020	$0.02
-	PCB pick and place	1	$1.000	$1.00
-	Automated PCB test	1	$0.500	$0.50
-	PCB rework	0.1	$2.000	$0.20
-	PCB yield loss	0.01	$10.000	$0.10
-	Automated assembly	1	$2.000	$2.00
-	Manual assembly	1	$2.000	$2.00
-	Automated testing	1	$1.000	$1.00
-	Rework	0.01	$10.000	$0.10
-	Packing foam	1	$0.100	$0.10
-	Wrapper	1	$0.050	$0.05
-	Instruction book	1	$0.200	$0.20
-	Cardboard box	1	$0.100	$0.10
	Total			**$29.00**

Table 4. Ink cartridge manufacturing cost

part ref.	item	quantity	unit cost	cost
-	Upper molding	1	$0.020	$0.02
-	Lower molding	1	$0.030	$0.03
-	Elastomeric seal molding	1	$0.020	$0.02

Table 4. Ink cartridge manufacturing cost

part ref	item	quantity	unit cost	cost
-	QA chip	1	$0.100	$0.10
-	Cyan ink (liter)	0.1	$4.000	$0.40
-	Magenta ink (liter)	0.1	$4.000	$0.40
-	Yellow ink (liter)	0.1	$4.000	$0.40
-	Black ink (liter)	0.3	$3.000	$0.90
-	Front pressure sensitive label	1	$0.020	$0.02
-	Airtight plastic bag	1	$0.020	$0.02
-	Instruction sheet	1	$0.010	$0.01
-	Cardboard box	1	$0.040	$0.04
-	Automated assembly	1	$0.050	$0.05
	Total			**$2.41**

6 Printer Control Protocol

This section describes the printer control protocol used between a host and CePrint. It includes control and status handling as well as the actual page description.

6.1 Control and Status

The printer control protocol defines the format and meaning of messages exchanged by the host processor and printer. The control protocol is defined independently of the transport protocol between the host processor and the printer, since the transport protocol depends on the exact nature of the connection.

Each message consists of a 16-bit message code, followed by message-specific data which may be of fixed or variable size.

All integers contained in messages are encoded in big-endian byte order.

Table 5 defines command messages sent by the host processor to the printer.

Table 5. Printer command messages

command message	message code	description
reset printer	1	Resets the printer to an idle state (i.e. ready and not printing).
get printer status	2	Gets the current printer status.
start document	3	Starts a new document.
start page	4	Starts the description of a new output page.
page band	5	Describes a band of the current output page.
end page	6	Ends the description of the current output page.
end document	7	Ends the current document.

The *reset printer* command can be used to reset the printer to clear an error condition, and to abort printing.

The *start document* command is used to indicate the start of a new document. This resets the printer's page count, which is used in the double-sided version to identify odd and even pages. The *end document* command is simply used to indicate the end of the document.

The description of an output page consists of a page header which describes the size and resolution of the page, followed by one or more page bands which describe the actual page content. The page header is transmitted to the printer in the *start page* command. Each page band is transmitted to the printer in a *page band* command. The last page band is followed by an *end page* command. The page description is described in detail in Table 6.2.

Table 6 defines response messages sent by the printer to the host processor.

Table 6. Printer response messages

response message	message code	description
printer status	8	Contains the current printer status (as defined in Table 7).
page error	9	Contains the most recent page error code (as defined in Table 8).

A *printer status* message is normally sent in response to a *get printer status* command. However, the nature of the connection between the host processor and the printer may allow the printer to send unsolicited status messages to the host processor. Unsolicited status messages allow timely reporting of printer exceptions to the host processor, and thereby to the user, without requiring the host processor to poll the printer on a frequent basis.

A *page error* message is sent in response to each *start page*, *page band* and *end page* command.

Table 7 defines the format of the 16-bit printer status contained in the *printer status* message.

Table 7. Printer status format

field	bit	description
ready	0	The printer is ready to receive a page.
printing	1	The printer is printing.
error	2	The printer is in an error state.
paper tray missing	3	The paper tray is missing.
paper tray empty	4	The paper tray is empty.
ink cartridge missing	5	The ink cartridge is missing.
ink cartridge empty	6	The ink cartridge is empty.
ink cartridge error	7	The ink cartridge is in an error state.
(reserved)	8-15	Reserved for future use.

Table 8 defines page error codes which may be returned in a *page error* message.

Table 8. Page error codes

error code	value	description
no error	0	No error.
bad signature	1	The signature is not recognized.
bad version	2	The version is not supported.
bad parameter	3	A parameter is incorrect.

6.2 PAGE DESCRIPTION

CePrint reproduces black at full dot resolution (1600 dpi), but reproduces contone color at a somewhat lower resolution using halftoning. The page description is therefore divided

into a black layer and a contone layer. The black layer is defined to composite *over* the contone layer.

The black layer consists of a bitmap containing a 1-bit *opacity* for each pixel. This black layer *matte* has a resolution which is an integer factor of the printer's dot resolution. The highest supported resolution is 1600 dpi, i.e. the printer's full dot resolution.

The contone layer consists of a bitmap containing a 32-bit CMYK *color* for each pixel. This contone image has a resolution which is an integer factor of the printer's dot resolution. The highest supported resolution is 267 ppi, i.e. one-sixth the printer's dot resolution.

The contone resolution is also typically an integer factor of the black resolution, to simplify calculations in the printer driver. This is not a requirement, however.

The black layer and the contone layer are both in compressed form for efficient transmission over the low-speed connection to the printer.

6.2.1 Page Structure

CePrint prints with full edge bleed using an 8.5" printhead. It imposes no margins and so has a printable page area which corresponds to the size of its paper (A4 or Letter).

The target page size is constrained by the printable page area, less the explicit (target) left and top margins specified in the page description.

These relationships are illustrated in Figure 20.

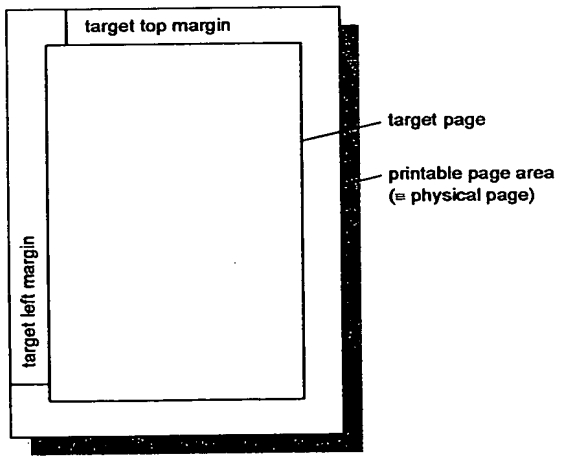


Figure 20. Page structure

6.2.2 Page Description Format

Apart from being implicitly defined in relation to the printable page area, each page description is complete and self-contained. There is no data transmitted to the printer separately from the page description to which the page description refers.

The page description consists of a page header which describes the size and resolution of the page, followed by one or more page bands which describe the actual page content.

Table 9 shows the format of the page header.

Table 9. Page header format

field	format	description
signature	16-bit integer	Page header format signature.
version	16-bit integer	Page header format version number.
structure size	16-bit integer	Size of page header.
target resolution (dpi)	16-bit integer	Resolution of target page. This is always 1600 for CePrint.
target page width	16-bit integer	Width of target page, in dots.
target page height	16-bit integer	Height of target page, in dots.
target left margin	16-bit integer	Width of target left margin, in dots.
target top margin	16-bit integer	Height of target top margin, in dots.
black scale factor	16-bit integer	Scale factor from black resolution to target resolution (must be 2 or greater).
black page width	16-bit integer	Width of black page, in black pixels.
black page height	16-bit integer	Height of black page, in black pixels.
contone scale factor	16-bit integer	Scale factor from contone resolution to target resolution (must be 6 or greater).
contone page width	16-bit integer	Width of contone page, in contone pixels.
contone page height	16-bit integer	Height of contone page, in contone pixels.

The page header contains a signature and version which allow the printer to identify the page header format. If the signature and/or version are missing or incompatible with the printer, then the printer can reject the page.

The page header defines the resolution and size of the target page. The black and contone layers are clipped to the target page if necessary. This happens whenever the black or contone scale factors are not factors of the target page width or height.

The target left and top margins define the positioning of the target page within the printable page area.

The black layer parameters define the pixel size of the black layer, and its integer scale factor to the target resolution.

The contone layer parameters define the pixel size of the contone layer, and its integer scale factor to the target resolution.

Table 10 shows the format of the page band header.

Table 10. Page band header format

field	format	description
signature	16-bit integer	Page band header format signature.
version	16-bit integer	Page band header format version number.
structure size	16-bit integer	Size of page band header.
black band height	16-bit integer	Height of black band, in black pixels.
black band data size	32-bit integer	Size of black band data, in bytes.

Table 10. Page band header format

field	format	description
contone band height	16-bit integer	Height of contone band, in contone pixels.
contone band data size	32-bit integer	Size of contone band data, in bytes.

The black layer parameters define the height of the black band, and the size of its compressed band data. The variable-size black band data follows the fixed-size parts of the page band header.

The contone layer parameters define the height of the contone band, and the size of its compressed page data. The variable-size contone band data follows the variable-size black band data.

Table 11 shows the format of the variable-size compressed band data which follows the page band header.

Table 11. Page band data format

field	format	description
black band data	EDRL bytestream	Compressed bi-level black band data.
contone band data	JPEG bytestream	Compressed contone CMYK band data.

The variable-size black band data and the variable-size contone band data are aligned to 8-byte boundaries. The size of the required padding is included in the size of the fixed-size part of the page band header structure and the variable-size black band data.

The entire page description has a target size of less than 3MB, and a maximum size of 6MB, in accordance with page buffer memory in the printer.

The following sections describe the format of the compressed black layer and the compressed contone layer.

6.2.3 Bi-level Black Layer Compression

6.2.3.1 Group 3 and 4 Facsimile Compression

The Group 3 Facsimile compression algorithm [1] losslessly compresses bi-level data for transmission over slow and noisy telephone lines. The bi-level data represents scanned black text and graphics on a white background, and the algorithm is tuned for this class of images (it is explicitly not tuned, for example, for *halftoned* bi-level images). The *1D* Group 3 algorithm runlength-encodes each scanline and then Huffman-encodes the resulting runlengths. Runlengths in the range 0 to 63 are coded with *terminating* codes. Runlengths in the range 64 to 2623 are coded with *make-up* codes, each representing a multiple of 64, followed by a terminating code. Runlengths exceeding 2623 are coded with multiple make-up codes followed by a terminating code. The Huffman tables are fixed, but are separately tuned for black and white runs (except for make-up codes above 1728, which are common). When possible, the *2D* Group 3 algorithm encodes a scanline as a set of short edge deltas (0, ±1, ±2, ±3) with reference to the previous scanline. The delta symbols are entropy-encoded (so that the zero delta symbol is only one bit long etc.) Edges within a 2D-encoded line which can't be delta-encoded are runlength-encoded, and are identified by a prefix. 1D- and 2D-encoded lines are marked differently. 1D-encoded lines are generated at regular intervals, whether actually required or not, to ensure that the

decoder can recover from line noise with minimal image degradation. 2D Group 3 achieves compression ratios of up to 6:1 [16].

The Group 4 Facsimile algorithm [1] losslessly compresses bi-level data for transmission over *error-free* communications lines (i.e. the lines are truly error-free, or error-correction is done at a lower protocol level). The Group 4 algorithm is based on the 2D Group 3 algorithm, with the essential modification that since transmission is assumed to be error-free, 1D-encoded lines are no longer generated at regular intervals as an aid to error-recovery. Group 4 achieves compression ratios ranging from 20:1 to 60:1 for the CCITT set of test images [16].

The design goals and performance of the Group 4 compression algorithm qualify it as a compression algorithm for the bi-level black layer. However, its Huffman tables are tuned to a lower scanning resolution (100-400 dpi), and it encodes runlengths exceeding 2623 awkwardly. At 800 dpi, our maximum runlength is currently 6400. Although a Group 4 decoder core might be available for use in the printer controller chip (Section 7), it might not handle runlengths exceeding those normally encountered in 400 dpi facsimile applications, and so would require modification.

Since most of the benefit of Group 4 comes from the delta-encoding, a simpler algorithm based on delta-encoding alone is likely to meet our requirements. This approach is described in detail below.

6.2.3.2 Bi-Level Edge Delta and Runlength (EDRL) Compression Format

The *edge delta and runlength* (EDRL) compression format is based loosely on the Group 4 compression format and its precursors [1,18].

EDRL uses three kinds of symbols, appropriately entropy-coded. These are *create edge*, *kill edge*, and *edge delta*. Each line is coded with reference to its predecessor. The predecessor of the first line is defined to a line of white. Each line is defined to start off white. If a line actually starts of black (the less likely situation), then it must define a black edge at offset zero. Each line must define an edge at its left-hand end, i.e. at offset *page width*.

An edge can be coded with reference to an edge in the previous line if there is an edge within the maximum delta range with the same sense (white-to-black or black-to-white). This uses one of the *edge delta* codes. The shorter and likelier deltas have the shorter codes. The maximum delta range (±2) is chosen to match the distribution of deltas for typical glyph edges. This distribution is mostly independent of point size. A typical example is given in Table 12.

Table 12. Edge delta distribution for 10 point Times at 800 dpi

delta	probability
0	65%
1	23%
2	7%
≥ 3	5%

An edge can also be coded using the length of the run from the previous edge in the same line. This uses one of the *create edge* codes for short (7-bit) and long (13-bit) runlengths. For simplicity, and unlike Group 4, runlengths are not entropy-coded. In order to keep edge deltas implicitly synchronized with edges in the previous line, each unused edge in

the previous line is 'killed' when passed in the current line. This uses the *kill edge* code. The *end-of-page* code signals the end of the page to the decoder.

Note that 7-bit and 13-bit runlengths are specifically chosen to support 800 dpi A4/Letter pages. Longer runlengths could be supported without significant impact on compression performance. For example, if supporting 1600 dpi compression, the runlengths should be at least 8-bit and 14-bit respectively. A general-purpose choice might be 8-bit and 16-bit, thus supporting up to 40" wide 1600 dpi pages.

The full set of codes is defined in Table 13. Note that there is no *end-of-line* code. The decoder uses the *page width* to detect the end of the line. The lengths of the codes are ordered by the relative probabilities of the codes' occurrence.

Table 13. EDRL codewords

code	encoding	suffix	description
Δ0	1	-	don't move corresponding edge
Δ+1	010	-	move corresponding edge +1
Δ–1	011	-	move corresponding edge –1
Δ+2	00010	-	move corresponding edge +2
Δ–2	00011	-	move corresponding edge –2
kill edge	0010	-	kill corresponding edge
create near edge	0011	7-bit RL	create edge from short runlength (RL)
create far edge	00001	13-bit RL	create edge from long runlength (RL)
end-of-page (EOP)	000001	-	end-of-page marker

Figure 21 shows a simple encoding example. Note that the common situation of an all-white line following another all-white line is encoded using a single bit (Δ0), and an all-black line following another all-black line is encoded using two bits (Δ0, Δ0).

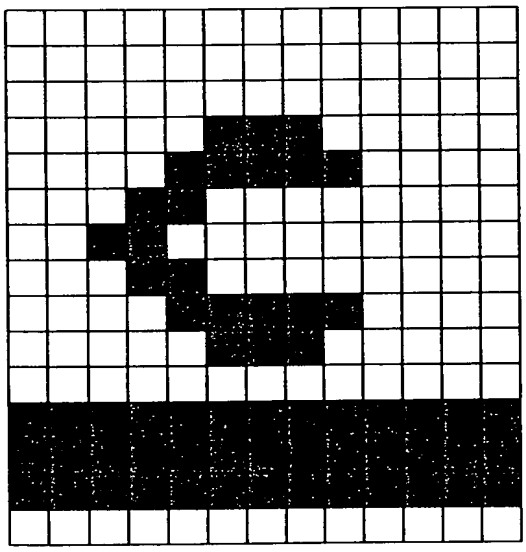


Figure 21. EDRL encoding example

Note that the foregoing describes the compression *format*, not the compression algorithm *per se*. A variety of equivalent encodings can be produced for the same image, some more compact than others. For example, a pure runlength encoding conforms to the compression format. The goal of the compression algorithm is to discover a good, if not the best, encoding for a given image.

The following is a simple algorithm for producing the EDRL encoding of a line with reference to its predecessor.

```
#define SHORT_RUN_PRECISION7  // precision of short run
#define LONG_RUN_PRECISION13  // precision of long run

EDRL_CompressLine
(
   Byte prevLine[],           // previous (reference) bi-level line
   Byte currLine[],           // current (coding) bi-level line
   int lineLen,               // line length
   BITSTREAM s                // output (compressed) bitstream
)
   int prevEdge = 0           // current edge offset in previous line
   int currEdge = 0           // current edge offset in current line
   int codedEdge = currEdge   // most recent coded (output) edge
   int prevColor = 0          // current color in previous line (0 = white)
   int currColor = 0          // current color in current line
   int prevRun                // current run in previous line
   int currRun                // current run in current line
   bool bUpdatePrevEdge = true// force first edge update
   bool bUpdateCurrEdge = true// force first edge update

   while (codedEdge < lineLen)

      // possibly update current edge in previous line
      if (bUpdatePrevEdge)
         if (prevEdge < lineLen)
            prevRun = GetRun(prevLine, prevEdge, lineLen, prevColor)
         else
            prevRun = 0
         prevEdge += prevRun
         prevColor = !prevColor
         bUpdatePrevEdge = false

      // possibly update current edge in current line
      if (bUpdateCurrEdge)
         if (currEdge < lineLen)
            currRun = GetRun(currLine, currEdge, lineLen, currColor)
         else
            currRun = 0
         currEdge += currRun
         currColor = !currColor
         bUpdateCurrEdge = false

      // output delta whenever possible, i.e. when
      // edge senses match, and delta is small enough
      if (prevColor == currColor)
         delta = currEdge - prevEdge
         if (abs(delta) <= MAX_DELTA)
            PutCode(s, EDGE_DELTA0 + delta)
            codedEdge = currEdge
            bUpdatePrevEdge = true
            bUpdateCurrEdge = true
            continue
```

```
// kill unmatched edge in previous line
if (prevEdge <= currEdge)
   PutCode(s, KILL_EDGE)
   bUpdatePrevEdge = true

// create unmatched edge in current line
if (currEdge <= prevEdge)
   PutCode(s, CREATE_EDGE)
   if (currRun < 128)
      PutCode(s, CREATE_NEAR_EDGE)
      PutBits(currRun, SHORT_RUN_PRECISION)
   else
      PutCode(s, CREATE_FAR_EDGE)
      PutBits(currRun, LONG_RUN_PRECISION)
   codedEdge = currEdge
   bUpdateCurrEdge = true
```

Figure 22. Algorithm for EDRL-compressing a bi-level line

Note that the algorithm is blind to *actual* edge continuity between lines, and may in fact match the "wrong" edges between two lines. Happily the compression format has nothing to say about this, since it decodes correctly, and it is difficult for a "wrong" match to have a detrimental effect on the compression ratio.

For completeness the corresponding decompression algorithm is given below. It forms the core of the EDRL Expander unit in the printer controller chip (Section 7).

```
EDRL_DecompressLine
(
   BITSTREAM s,                  // input (compressed) bitstream
   Byte prevLine[],              // previous (reference) bi-level line
   Byte currLine[],              // current (coding) bi-level line
   int lineLen                   // line length
)
   int prevEdge = 0              // current edge offset in previous line
   int currEdge = 0              // current edge offset in current line
   int prevColor = 0             // current color in previous line (0 = white)
   int currColor = 0             // current color in current line

   while (currEdge < lineLen)

      code = GetCode(s)
      switch (code)
         case EDGE_DELTA_MINUS2:
         case EDGE_DELTA_MINUS1:
         case EDGE_DELTA_0:
         case EDGE_DELTA_PLUS1:
         case EDGE_DELTA_PLUS2:
            // create edge from delta
            int delta = code - EDGE_DELTA_0
            int run = prevEdge + delta - currEdge
            FillBitRun(currLine, currEdge, currColor, run)
            currEdge += run
            currColor = !currColor
            prevEdge += GetRun(prevLine, prevEdge, lineLen, prevColor)
            prevColor = !prevColor

         case KILL_EDGE:
            // discard unused reference edge
            prevEdge += GetRun(prevLine, prevEdge, lineLen, prevColor)
```

```
        prevColor = !prevColor

    case CREATE_NEAR_EDGE:
    case CREATE_FAR_EDGE:
        // create edge explicitly
        int run
        if (code == CREATE_NEAR_EDGE)
            run = GetBits(s, SHORT_RUN_PRECISION)
        else
            run = GetBits(s, LONG_RUN_PRECISION)
        FillBitRun(currLine, currEdge, currColor, run)
        currColor = !currColor
        currEdge += run
```

Figure 23. Algorithm for decompressing an EDRL-compressed bi-level line

6.2.3.3 EDRL Compression Performance

Table 14 shows the compression performance of Group 4 and EDRL on the CCITT test documents used to select the Group 4 algorithm. Each document represents a single page scanned at 400 dpi. Group 4's superior performance is due to its entropy-coded run-lengths, tuned to 400 dpi features.

Table 14. Group 4 and EDRL compression performance on standard CCITTT documents at 400 dpi

CCITT document number	Group 4 compression ratio	EDRL compression ratio
1	29.1	21.6
2	49.9	41.3
3	17.9	14.1
4	7.3	5.5
5	15.8	12.4
6	31.0	25.5
7	7.4	5.3
8	26.7	23.4

Magazine text is typically typeset in a typeface with serifs (such as Times) at a point size of 10. At this size an A4/Letter page holds up to 14,000 characters, though a typical magazine page holds only about 7,000 characters. Text is seldom typeset at a point size smaller than 5. At 800 dpi, text cannot be meaningfully rendered at a point size lower than 2 using a standard typeface. Table 15 illustrates the legibility of various point sizes.

Table 15. Text at different point sizes

point size	sample text (in Times)
2	
3	
4	
5	The quick brown fox jumps over the lazy dog.
6	The quick brown fox jumps over the lazy dog.
7	The quick brown fox jumps over the lazy dog.
8	The quick brown fox jumps over the lazy dog.

Table 15. Text at different point sizes

point size	sample text (in Times)
9	The quick brown fox jumps over the lazy dog.
10	The quick brown fox jumps over the lazy dog.

Table 16 shows Group 4 and EDRL compression performance on pages of text of varying point sizes, rendered at 800 dpi. Note that EDRL achieves the required compression ratio of 2.5 for an *entire page* of text typeset at a point size of 3. The distribution of characters on the test pages is based on English-language statistics [15].

Table 16. Group 4 and EDRL compression performance on text at 800 dpi

point size	characters/ A4 page	Group 4 compression ratio	EDRL compression ratio
2	340,000	2.3	1.7
3	170,000	3.2	2.5
4	86,000	4.7	3.8
5	59,000	5.5	4.9
6	41,000	6.5	6.1
7	28,000	7.7	7.4
8	21,000	9.1	9.0
9	17,000	10.2	10.4
10	14,000	10.9	11.3
11	12,000	11.5	12.4
12	8,900	13.5	14.8
13	8,200	13.5	15.0
14	7,000	14.6	16.6
15	5,800	16.1	18.5
20	3,400	19.8	23.9

For a point size of 9 or greater, EDRL slightly outperforms Group 4, simply because Group 4's runlength codes are tuned to 400 dpi.

These compression results bear out the observation that entropy-encoded runlengths contribute much less to compression than 2D encoding, unless the data is poorly correlated vertically, such as in the case of very small characters.

6.2.4 Contone Layer Compression

6.2.4.1 JPEG Compression

The JPEG compression algorithm [5] lossily compresses a contone image at a specified quality level. It introduces imperceptible image degradation at compression ratios below 5:1, and negligible image degradation at compression ratios below 10:1 [17].

JPEG typically first transforms the image into a color space which separates luminance and chrominance into separate color channels. This allows the chrominance channels to be subsampled without appreciable loss because of the human visual system's relatively greater sensitivity to luminance than chrominance. After this first step, each color channel is compressed separately.

The image is divided into 8×8 pixel blocks. Each block is then transformed into the frequency domain via a discrete cosine transform (DCT). This transformation has the effect of concentrating image energy in relatively lower-frequency coefficients, which allows higher-frequency coefficients to be more crudely quantized. This quantization is the principal source of compression in JPEG. Further compression is achieved by ordering coefficients by frequency to maximize the likelihood of adjacent zero coefficients, and then runlength-encoding runs of zeroes. Finally, the runlengths and non-zero frequency coefficients are entropy coded. Decompression is the inverse process of compression.

6.2.4.2 CMYK Contone JPEG Compression Format

The CMYK contone layer is compressed to an interleaved color JPEG bytestream. The interleaving is required for space-efficient decompression in the printer, but may restrict the decoder to two sets of Huffman tables rather than four (i.e. one per color channel) [17]. If luminance and chrominance are separated, then the luminance channels can share one set of tables, and the chrominance channels the other set.

If luminance/chrominance separation is deemed necessary, either for the purposes of table sharing or for chrominance subsampling, then CMY is converted to YCrCb and Cr and Cb are duly subsampled. K is treated as a luminance channel and is not subsampled.

The JPEG bytestream is complete and self-contained. It contains all data required for decompression, including quantization and Huffman tables.

7 Printer Controller

7.1 PRINTER CONTROLLER ARCHITECTURE

The printer controller consists of the CePrint central processor (CCP) chip, a 64MBit RDRAM, and the master QA chip.

The CCP contains a general-purpose processor and a set of purpose-specific functional units controlled by the processor via the processor bus, as shown in Figure 24. Only three functional units are non-standard - the EDRL expander, the halftoner/compositor, and the printhead interface which controls the Memjet printhead.

Software running on the processor coordinates the various functional units to receive, expand and print pages. This is described in the next section.

The various functional units of the CCP are described in subsequent sections.

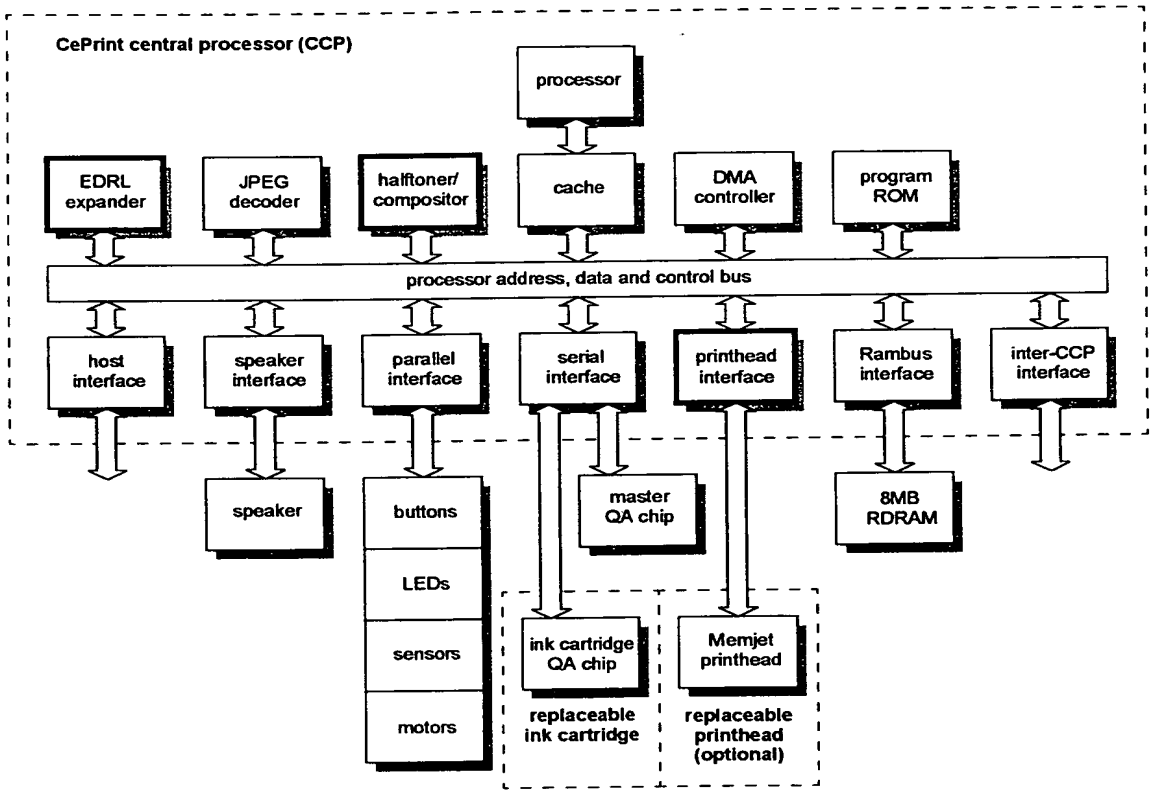


Figure 24. Printer controller architecture

7.2 PAGE EXPANSION AND PRINTING

Page expansion and printing proceeds as follows. A page description is received from the host via the host interface and is stored in main memory. 6MB of main memory is dedicated to page storage. This can hold two pages each not exceeding 3MB, or one page up to 6MB. If the host generates pages not exceeding 3MB, then the printer operates in stream-

ing mode - i.e. it prints one page while receiving the next. If the host generates pages exceeding 3MB, then the printer operates in single-page mode - i.e. it receives each page and prints it before receiving the next. If the host generates pages exceeding 6MB then they are rejected by the printer. In practice the printer driver prevents this from happening.

A page consists of two parts - the bi-level black layer, and the contone layer. These are compressed in distinct formats - the bi-level black layer in EDRL format, the contone layer in JPEG format. The first stage of page expansion consists of decompressing the two layers in parallel. The bi-level layer is decompressed by the EDRL expander unit, the contone layer by the JPEG decoder.

The second stage of page expansion consists of halftoning the contone CMYK data to bi-level CMYK, and then compositing the bi-level black layer over the bi-level CMYK layer. The halftoning and compositing is carried out by the halftoner/compositor unit.

Finally, the composited bi-level CMYK image is printed via the printhead interface unit, which controls the Memjet printhead.

Because the Memjet printhead prints at high speed, the paper must move past the printhead at a constant velocity. If the paper is stopped because data can't be fed to the printhead fast enough, then visible printing irregularities will occur. It is therefore important to transfer bi-level CMYK data to the printhead interface at the required rate.

A fully-expanded 1600 dpi bi-level CMYK page has an image size of 119MB. Because it is impractical to store an expanded page in printer memory, each page is expanded in real time during printing. Thus the various stages of page expansion and printing are pipelined. The page expansion and printing data flow is described in Table 17. The aggregate traffic to/from main memory of 182MB/s is well within the capabilities of current technologies such as Rambus.

Table 17. Page expansion and printing data flow

process	input	input window	output	output window	input rate	output rate
receive contone	-	-	JPEG stream	1	- -	1.5MB/s 3.5Mp/s
receive bi-level	-	-	EDRL stream	1	- -	1.5MB/s 31Mp/s
decompress contone	JPEG stream	-	32-bit CMYK	8	1.5MB/s 3.5Mp/s	13MB/s 3.5Mp/s
decompress bi-level	EDRL stream	-	1-bit K	1	1.5MB/s 31Mp/s[a]	15MB/s 124Mp/s
halftone	32-bit CMYK	1	-[b]	-	13MB/s 3.5Mp/s[c]	-
composite	1-bit K	1	4-bit CMYK	1	15MB/s 124Mp/s	60MB/s 124Mp/s
print	4-bit CMYK	24, 1[d]	-	-	60MB/s 124Mp/s	- -
					91MB/s	91MB/s
						182MB/s

a. 800 dpi ⇒ 1600 dpi (2 × 2 expansion).
b. halftone combines with composite, so there is no external data flow between them
c. 267 ppi ⇒ 1600 dpi (6 × 6 expansion).
d. Needs a window of 24 lines, but only advances 1 line.

Each stage communicates with the next via a shared FIFO in main memory. Each FIFO is organized into lines, and the minimum size (in lines) of each FIFO is designed to accommodate the output window (in lines) of the producer and the input window (in lines) of the consumer. The inter-stage main memory buffers are described in Table 18. The aggregate buffer space usage of 6.3MB leaves 1.7MB free for program code and scratch memory (out of the 8MB available).

Table 18. Page expansion and printing main memory buffers

buffer	organization and line size	number of lines	buffer size
compressed page buffer	byte stream (one or two pages) -	-	6MB
contone CMYK buffer	32-bit interleaved CMYK (267 ppi x 8.5" x 32 = 8.9KB)	8 x 2 = 16	142KB
bi-level K buffer	1-bit K (1600 dpi x 8.5" x 1 = 1.7B)	1 x 2 = 2	3KB
bi-level CMYK buffer	4-bit planar odd/even CMYK (1600 dpi x 8.5" x 4 = 6.6KB)	24 + 1 = 25	166KB
			6.3MB

The overall data flow, including FIFOs, is illustrated in Figure 25.

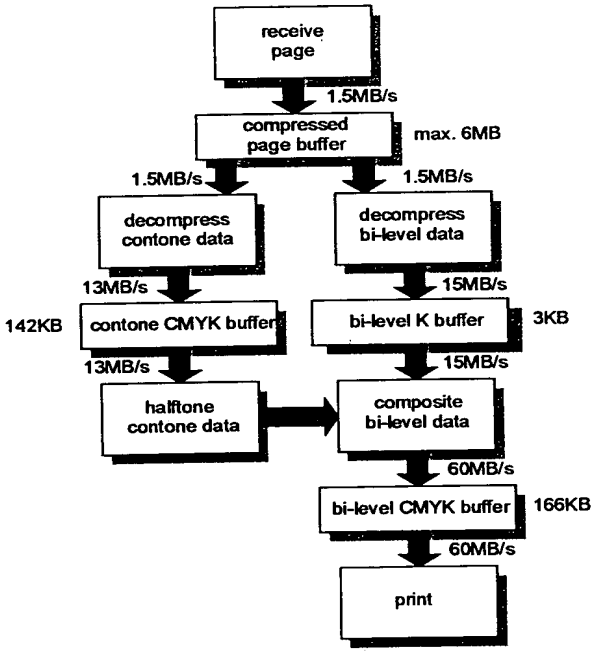


Figure 25. Page expansion and printing data flow summary

Contone page decompression is carried out by the JPEG decoder. Bi-level page decompression is carried out by the EDRL expander. Halftoning and compositing is carried out by the halftoner/compositor unit. These functional units are described in the following sections.

7.2.1 DMA Approach

Each functional unit contains one or more on-chip input and/or output FIFOs. Each FIFO is allocated a separate channel in the multi-channel DMA controller. The DMA controller handles single-address rather than double-address transfers, and so provides a separate request/acknowledge interface for each channel.

Each functional unit stalls gracefully whenever an input FIFO is exhausted or an output FIFO is filled.

The processor programs each DMA transfer. The DMA controller generates the address for each word of the transfer on request from the functional unit connected to the channel. The functional unit latches the word onto or off the data bus when its request is acknowledged by the DMA controller. The DMA controller interrupts the processor when the transfer is complete, thus allowing the processor to program another transfer on the same channel in a timely fashion.

In general the processor will program another transfer on a channel as soon as the corresponding main memory FIFO is available (i.e. non-empty for a read, non-full for a write).

The granularity of channel servicing implemented in the DMA controller depends somewhat on the latency of main memory.

7.2.2 EDRL Expander

The EDRL expander unit (EEU) (Figure 26) decompresses an EDRL-compressed bi-level image.

The input to the EEU is an EDRL bitstream. The output from the EEU is a set of bi-level image lines, scaled horizontally from the resolution of the expanded bi-level image by an integer scale factor to 1600 dpi.

Once started, the EEU proceeds until it detects an *end-of-page* code in the EDRL bitstream, or until it is explicitly stopped via its control register.

The EEU relies on an explicit page width to decode the bitstream. This must be written to the *page width* register prior to starting the EEU.

The scaling of the expanded bi-level image relies on an explicit scale factor. This must be written to the *scale factor* register prior to starting the EEU.

Table 19. EDRL expander control and configuration registers

register	width	description
start	1	Start the EEU.
stop	1	Stop the EEU.
page width	13	Page width used during decoding to detect end-of-line.
scale factor	4	Scale factor used during scaling of expanded image.

The EDRL compression format is described in Section 6.2.3.2. It represents a bi-level image in terms of its edges. Each edge in each line is coded relative to an edge in the previous line, or relative to the previous edge in the same line. No matter how it is coded, each edge is ultimately decoded to its distance from the previous edge in the same line. This distance, or runlength, is then decoded to the string of one bits or zero bits which rep-

resent the corresponding part of the image. The decompression algorithm is defined in Section 6.2.3.2.

The EEU consists of a bitstream decoder, a state machine, edge calculation logic, two runlength decoders, and a runlength (re)encoder.

The bitstream decoder decodes an entropy-coded codeword from the bitstream and passes it to the state machine. The state machine returns the size of the codeword to the bitstream decoder, which allows the decoder to advance to the next codeword. In the case of a *create edge* code, the state machine uses the bitstream decoder to extract the corresponding runlength from the bitstream. The state machine controls the edge calculation logic and runlength decoding/encoding as defined in Table 21.

The edge calculation logic is quite simple. The current edge offset in the previous (reference) and current (coding) lines are maintained in the reference edge register and edge register respectively. The runlength associated with a *create edge* code is output directly to the runlength decoder, and is added to the current edge. A *delta* code is translated into a runlength by adding the associated delta to the reference edge and subtracting the current edge. The generated runlength is output to the runlength decoder, and is added to the current edge. The next runlength is extracted from the runlength encoder and added to the reference edge. A *kill edge* code simply causes the current reference edge to be skipped. Again the next runlength is extracted from the runlength encoder and added to the reference edge.

Each time the edge calculation logic generates a runlength representing an edge, it is passed to the runlength decoder. While the runlength decoder decodes the run it generates a stall signal to the state machine. Since the runlength decoder is slower than the edge calculation logic, there's not much point in decoupling it. The expanded line accumulates in a line buffer large enough to hold an 8.5" 800 dpi line (850 bytes).

The previously expanded line is also buffered. It acts as a reference for the decoding of the current line. The previous line is re-encoded as runlengths on demand. This is less expensive than buffering the decoded runlengths of the previous line, since the worst case is one 13-bit runlength for each pixel (20KB at 1600 dpi). While the runlength encoder encodes the run it generates a stall signal to the state machine. The runlength encoder uses the page width to detect end-of-line. The (current) line buffer and the previous line buffer are concatenated and managed as a single FIFO to simplify the runlength encoder.

A second runlength decoder decodes the output runlength to a line buffer large enough to hold an 8.5" 1600 dpi line (1700 bytes). The runlength passed to this output runlength decoder is multiplied by the scale factor, so this decoder produces 1600 dpi lines. The line is output *scale factor* times through the output pixel FIFO. This achieves the required vertical scaling by simple line replication. The EEU could be designed with *edge smoothing* integrated into its image scaling. A simple smoothing scheme based on template-matching can be very effective [10]. This would require a multi-line buffer between the low-resolution runlength decoder and the smooth scaling unit, but would eliminate the high-resolution runlength decoder.

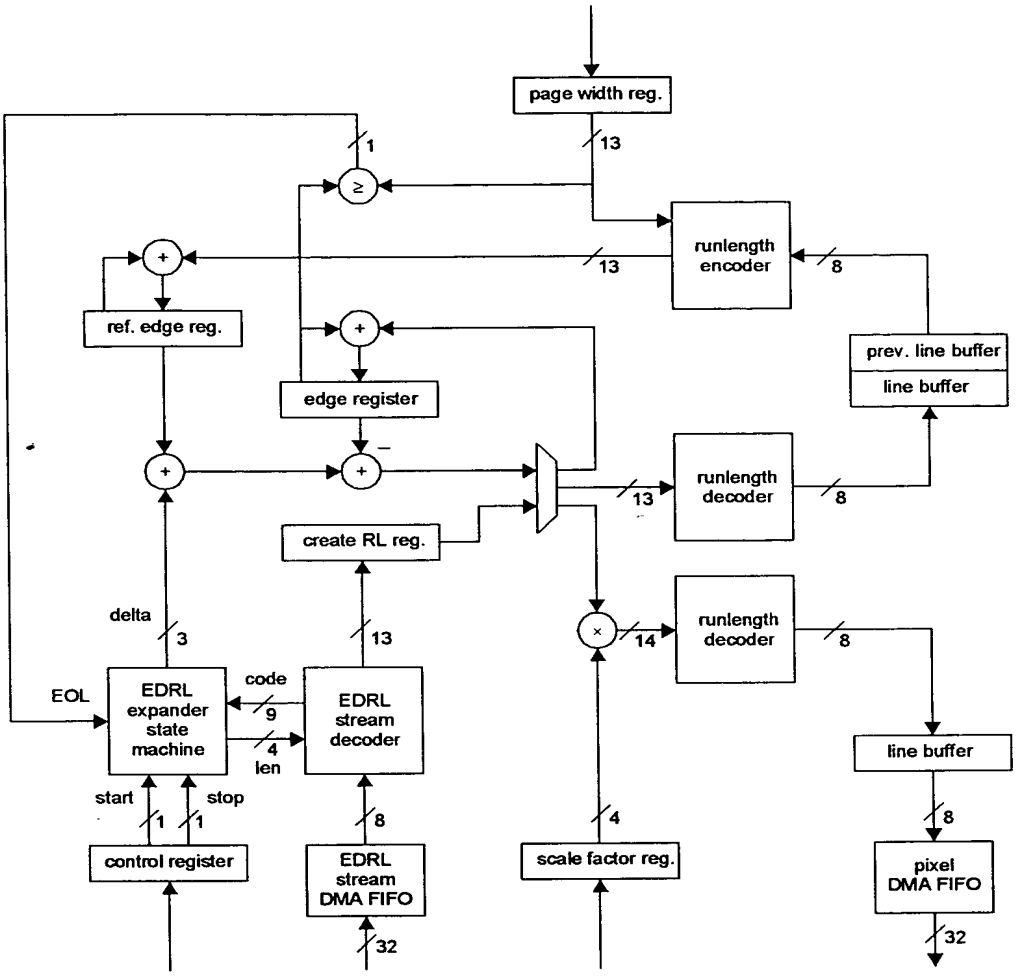


Figure 26. EDRL expander unit

7.2.2.1 *EDRL Stream Decoder*

The EDRL stream decoder decodes entropy-coded EDRL codewords in the input bitstream. It uses a two-byte input buffer viewed through a 16-bit barrel shifter whose left (most significant) edge is always aligned to a codeword boundary in the bitstream. The decoder connected to the barrel shifter decodes a codeword according to Table 20, and supplies the state machine with the corresponding code.

Table 20. EDRL stream codeword decoding table

input codeword bit pattern	output code	output code bit pattern
1xxx xxxx	Δ0	1 0000 0000
010x xxxx	Δ+1	0 1000 0000
011x xxxx	Δ−1	0 0100 0000
0010 xxxx	kill edge	0 0010 0000

Table 20. EDRL stream codeword decoding table

input codeword bit pattern[a]	output code	output code bit pattern
0011 xxxx	create near edge	0 0001 0000
0001 0xxx	Δ+2	0 0000 1000
0001 1xxx	Δ–2	0 0000 0100
0000 1xxx	create far edge	0 0000 0010
0000 01xx	end-of-page (EOP)	0 0000 0001

a. x = don't care

The state machine in turn outputs the length of the code. This is added, modulo-8, to the current codeword bit offset to yield the next codeword bit offset. The bit offset in turn controls the barrel shifter. If the codeword bit offset wraps, then the carry bit controls the latching of the next byte from the input FIFO. At this time byte 2 is latched to byte 1, and the FIFO output is latched to byte 2. It takes two cycles of length 8 to fill the input buffer. This is handled by starting states in the state machine.

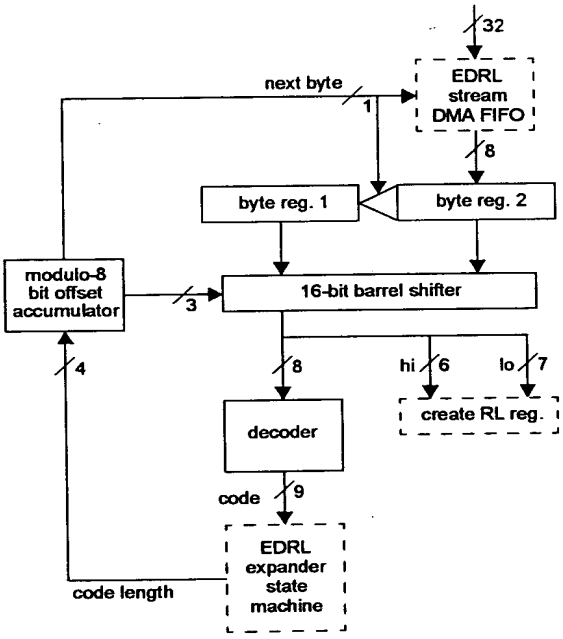


Figure 27. EDRL stream decoder

7.2.2.2 *EDRL Expander State Machine*

The EDRL expander state machine controls the edge calculation and runlength expansion logic in response to codes supplied by the EDRL stream decoder. It supplies the EDRL stream decoder with the length of the current codeword and supplies the edge calculation logic with the delta value associated with the current delta code. The state machine also responds to *start* and *stop* control signals from the control register, and the *end-of-line* (EOL) signal from the edge calculation logic.

The state machine also controls the multi-cycle fetch of the runlength associated with a *create edge* code.

Table 21. EDRL expander state machine

input signal	input code	current state	next state	code len	delta	actions
start	-	stopped	starting	8	-	-
-	-	starting	idle	8	-	-
stop	-	-	stopped	0	-	reset RL decoders and FIFOs
EOL	-	-	EOL 1	0	-	reset RL encoder; reset RL decoders; reset ref. edge and edge
-	-	EOL 1	idle			RL encoder ⇒ ref. RL; ref. edge += ref. RL
-	Δ0	idle	idle	1	0	RL = edge - ref. edge + delta; edge += RL; RL ⇒ RL decoder; RL encoder ⇒ ref. RL; ref. edge += ref. RL
-	Δ+1	idle	idle	2	+1	"
-	Δ–1	idle	idle	3	–1	"
-	Δ+2	idle	idle	4	+2	"
-	Δ–2	idle	idle	5	–2	"
-	kill edge	idle	idle	6	-	RL encoder ⇒ ref. RL; ref. edge += ref. RL
-	create near edge	idle	create RL lo 7	7	-	reset create RL
-	create far edge	idle	create RL hi 6	8	-	-
-	EOP	idle	stopped	8	-	-
-	-	create RL hi 6	create RL lo 7	6	-	latch create RL hi 6
-	-	create RL lo 7	create edge	7	-	latch create RL lo 7
-	-	create edge	idle	0	-	RL = create RL; edge += RL; RL ⇒ RL encoder

7.2.2.3 *Runlength Decoder*

The runlength decoder expands a runlength into a sequence of zero bits or one bits of the corresponding length in the output stream. The first run in a line is assumed to be white (color 0). Each run is assumed to be of the opposite color to its predecessor. If the first run is actually black (color 1), then it must be preceded by a zero-length white run. The runlength decoder keeps track of the current color internally.

The runlength decoder appends a maximum of 8 bits to the output stream every clock. Runlengths are typically not an integer multiple of 8, and so runs other than the first in an

image are typically not byte-aligned. The run decoder maintains, in the byte space register, the number of bits available in the byte currently being built. This is initialized to 8 at the beginning of decoding, and on the output of every byte.

The decoder starts outputting a run of bits as soon as the *next run* line latches a non-zero value into the runlength register. The decoder effectively stalls when the runlength register goes to zero.

A number of bits of the current color are shifted into the output byte register each clock. The current color is maintained in the 1-bit color register. The number of bits actually output is limited by the number of bits left in the runlength, and by the number of spare bits left in the output byte. The number of bits output is subtracted from the runlength and the byte space. When the runlength goes to zero it has been completely decoded, although the trailing bits of the run may still be in the output byte register, pending output. When the byte space goes to zero the output byte is full and is appended to the output stream.

The 16-bit barrel shifter, the output byte register and the color register together implement an 8-bit shift register which can be shifted multiple bit positions every clock, with the color as the serial input.

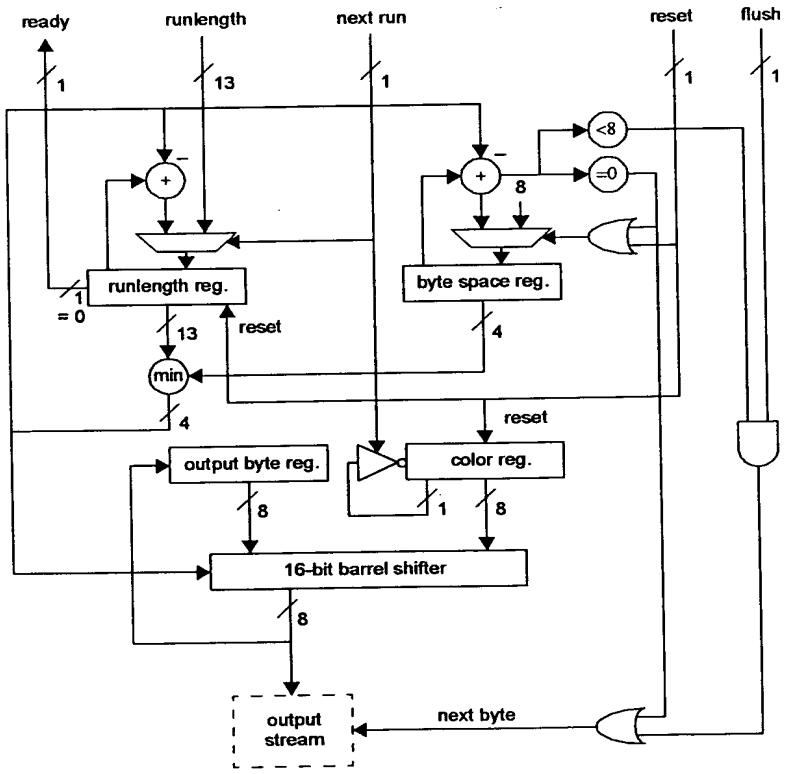


Figure 28. Runlength decoder

The external *reset* line is used to reset the runlength decoder at the start of a line. The external *next run* line is used to request the decoding of a new runlength. It is accompanied by a runlength on the external *runlength* lines. The *next run* line should not be set on the same clock as the *reset* line. Because *next run* inverts the current color, the reset of the color sets it to one, not zero. The external *flush* line is used to flush the last byte of the run,

if incomplete. It can be used on a line-by-line basis to yield byte-aligned lines, or on an image basis to yield a byte-aligned image.

The external *ready* line indicates whether the runlength decoder is ready to decode a runlength. It can be used to stall the external logic.

7.2.2.4 Runlength Encoder

The runlength encoder detects a run of zero or one bits in the input stream. The first run in a line is assumed to be white (color 0). Each run is assumed to be of the opposite color to its predecessor. If the first run is actually black (color 1), then the runlength encoder generates a zero-length white run at the start of the line. The runlength decoder keeps track of the current color internally.

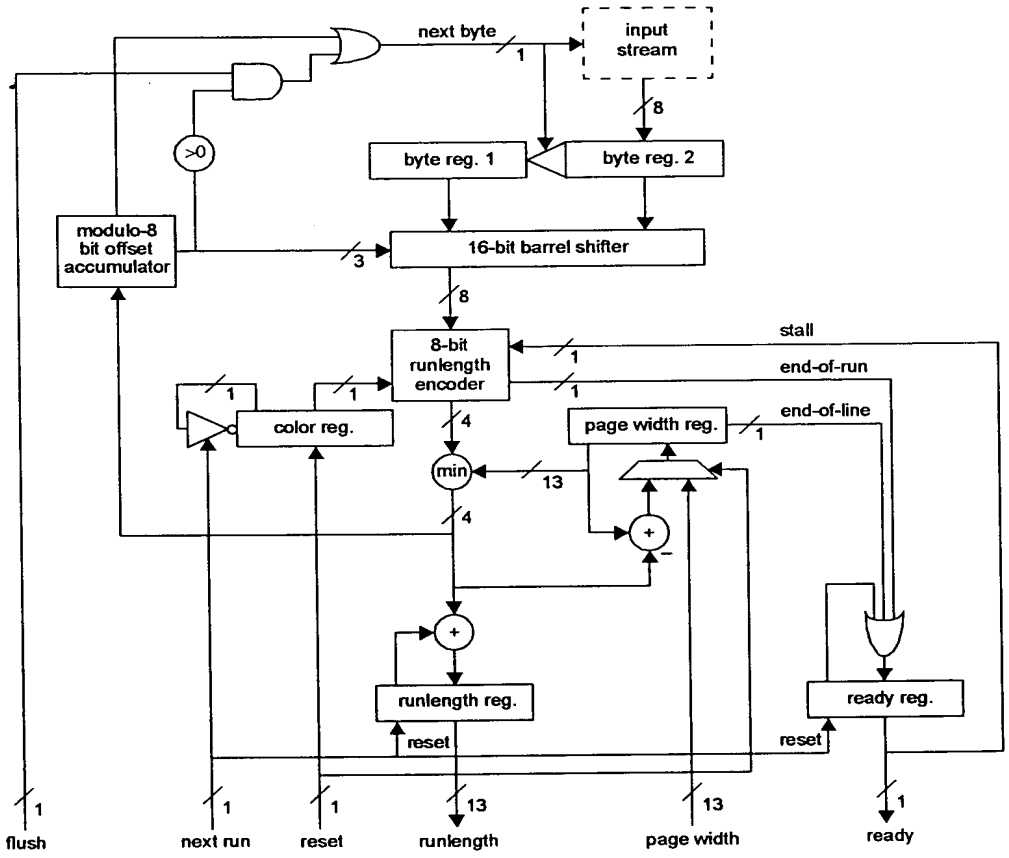


Figure 29. Runlength encoder

The runlength encoder reads a maximum of 8 bits from the input stream every clock. It uses a two-byte input buffer viewed through a 16-bit barrel shifter whose left (most significant) edge is always aligned to the current position in the bitstream. The encoder connected to the barrel shifter encodes an 8-bit (partial) runlength according to Table 22. The encoder uses the current color to recognize runs of the appropriate color.

The 8-bit runlength generated by the 8-bit runlength encoder is added to the value in the runlength register. When the 8-bit runlength encoder recognizes the end of the current run

it generates an end-of-run signal which is latched by the ready register. The output of the ready register indicates that the encoder has completed encoding the current runlength, accumulated in the runlength register. The output of the ready register is also used to stall the 8-bit runlength encoder. When stalled the 8-bit runlength encoder outputs a zero-length run and a zero end-of-run signal, effectively stalling the entire runlength encoder.

Table 22. 8-bit runlength encoder table

color	input	length	end-of-run
0	0000 0000	8	0
0	0000 0001	7	1
0	0000 001x	6	1
0	0000 01xx	5	1
0	0000 1xxx	4	1
0	0001 xxxx	3	1
0	001x xxxx	2	1
0	01xx xxxx	1	1
0	1xxx xxxx	0	1
1	1111 1111	8	0
1	1111 1110	7	1
1	1111 110x	6	1
1	1111 10xx	5	1
1	1111 0xxx	4	1
1	1110 xxxx	3	1
1	110x xxxx	2	1
1	10xx xxxx	1	1
1	0xxx xxxx	0	1

The output of the 8-bit runlength encoder is limited by the remaining page width. The actual 8-bit runlength is subtracted from the remaining page width, and is added to the modulo-8 bit position used to control the barrel shifter and clock the byte stream input.

The external *reset* line is used to reset the runlength encoder at the start of a line. It resets the current color and latches the *page width* into the page width register. The external *next run* line is used to request another runlength from the runlength encoder. It inverts the current color, and resets the runlength register and ready register. The external *flush* line is used to flush the last byte of the run, if incomplete. It can be used on a line-by-line basis to process byte-aligned lines, or on an image basis to process a byte-aligned image.

The external *ready* line indicates that the runlength encoder is ready to encode a runlength, and that the current runlength is available on the *runlength* lines. It can be used to stall the external logic.

7.2.2.5 Timing

The EEU has an output rate of 124M 1-bit black pixels/s. The core logic generates one runlength every clock. The runlength decoders and the runlength encoder generate/consume up to 8 pixels (bits) per clock. One runlength decoder and the runlength encoder operate at quarter resolution (800 dpi). The other runlength decoder operates at full resolution (1600 dpi).

A worst-case bi-level image consisting of a full page of 3 point text converts to approximately 6M runlengths at 800 dpi (the rendering resolution). At 1600 dpi (the horizontal output resolution) this gives an average runlength of about 20. Consequently about 40% of 8-pixel output bytes span two runs and so require two clocks instead of one. Output lines are replicated vertically to achieve a vertical resolution of 1600 dpi. When a line is being replicated rather than generated it has a perfect efficiency of 8 pixels per clock, thus the overhead is halved to 20%.

The full-resolution runlength decoder in the output stage of the EEU is slowest component in the EEU. The minimum clock speed of the EEU is therefore dictated by the output pixel rate of the EEU (124Mpixels/s), divided by the width of the runlength decoder (8), adjusted for its worst-case overhead (20%). This gives a minimum speed of about 22MHz.

7.2.3 JPEG Decoder

The JPEG decoder (Figure 30) decompresses a JPEG-compressed CMYK contone image.

The input to the JPEG decoder is a JPEG bitstream. The output from the JPEG decoder is a set of contone CMYK image lines.

When decompressing, the JPEG decoder writes its output in the form of 8×8 pixel blocks. These are sometimes converted to full-width lines via an *page width* × 8 strip buffer closely coupled with the codec. This would require a 67KB buffer. We instead use 8 parallel pixel FIFOs with shared bus access and 8 corresponding DMA channels, as shown in Figure 30.

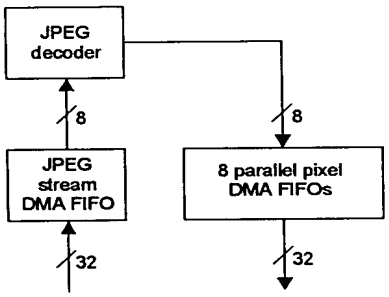


Figure 30. JPEG decoder

7.2.3.1 *Timing*

The JPEG decoder has an output rate of 3.5M 32-bit CMYK pixels/s. The required clock speed of the decoder depends on the design of the decoder.

7.2.4 Halftoner/Compositor

The halftoner/compositor unit (HCU) (Figure 32) combines the functions of halftoning the contone CMYK layer to bi-level CMYK, and compositing the black layer over the halftoned contone layer.

The input to the HCU is an expanded 267 ppi CMYK contone layer, and an expanded 1600 dpi black layer. The output from the HCU is a set of 1600 dpi bi-level CMYK image lines.

Once started, the HCU proceeds until it detects an *end-of-page* condition, or until it is explicitly stopped via its control register.

The HCU generates a page of dots of a specified width and length. The width and length must be written to the *page width* and *page length* registers prior to starting the HCU. The page width corresponds to the width of the printhead. The page length corresponds to the length of the target page.

The HCU generates target page data between specified left and right margins relative to the page width. The positions of the left and right margins must be written to the *left margin* and *right margin* registers prior to starting the HCU. The distance from the left margin to the right margin corresponds to the target page width.

The HCU consumes black and contone data according to specified black and contone page widths. These page widths must be written to the *black page width* and *contone page width* registers prior to starting the HCU. The HCU clips black and contone data to the target page width. This allows the black and contone page widths to exceed the target page width without requiring any special end-of-line logic at the input FIFO level.

The relationships between the page width, the black and contone page widths, and the margins are illustrated in Figure 31.

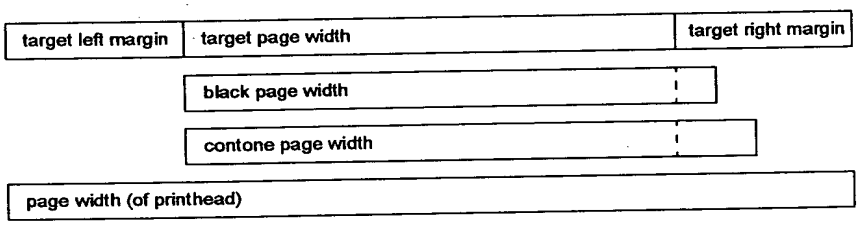


Figure 31. Relationships between page widths and margins

The HCU scales contone data to printer resolution both horizontally and vertically based on a specified scale factor. This scale factor must be written to the *contone scale factor* register prior to starting the HCU.

Table 23. Halftoner/compositor control and configuration registers

register	width	description
start	1	Start the HCU.
stop	1	Stop the HCU.
page width	14	Page width of printed page, in dots. This is the number of dots which have to be generated for each line.
left margin	14	Position of left margin, in dots.
right margin	14	Position of right margin, in dots.
page length	15	Page length of printed page, in dots. This is the number of lines which have to be generated for each page.
black page width	14	Page width of black layer, in dots. Used to detect the end of a black line.
contone page width	14	Page width of contone layer, in dots. Used to detect the end of a contone line.
contone scale factor	4	Scale factor used to scale contone data to bi-level resolution.

The consumer of the data produced by the HCU is the printhead interface. The printhead interface requires bi-level CMYK image data in *planar* format, i.e. with the color planes separated. Further, it also requires that even and odd pixels are separated. The output stage of the HCU therefore uses 8 parallel pixel FIFOs, one each for *even cyan*, *odd cyan*, *even magenta*, *odd magenta*, *even yellow*, *odd yellow*, *even black*, and *odd black*.

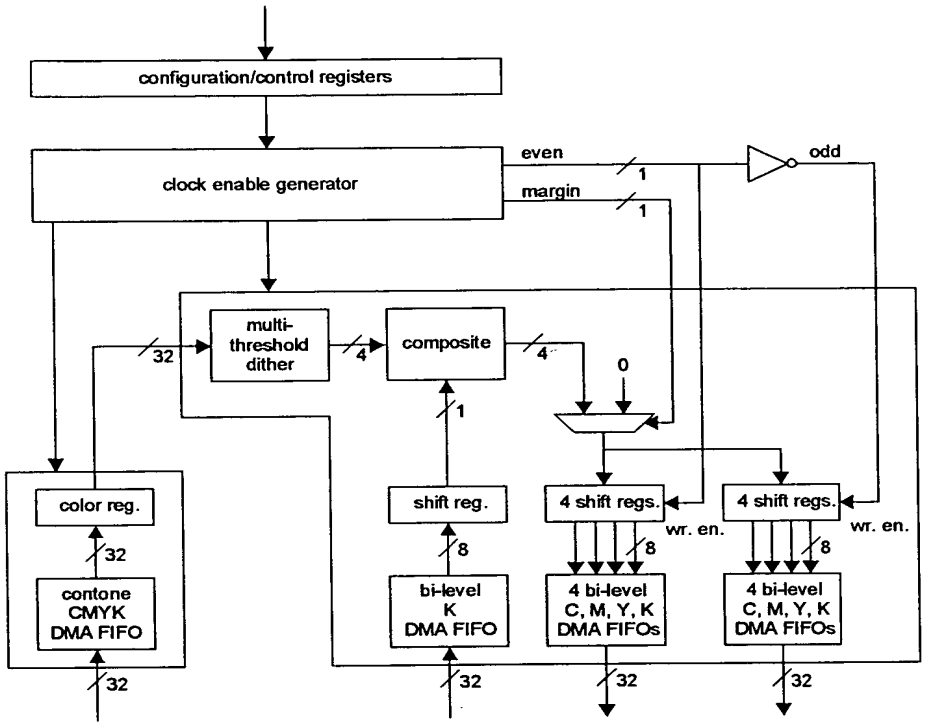


Figure 32. Halftoner/compositor unit

The input contone CMYK FIFO is a full 9KB line buffer. The line is used *contone scale factor* times to effect vertical up-scaling via line replication. FIFO write address wrapping is disabled until the start of the last use of the line. An alternative is to read the line from main memory *contone scale factor* times, increasing memory traffic by 44MB/s, but avoiding the need for the on-chip 9KB line buffer.

7.2.4.1 Multi-Threshold Dither

A general 256-layer *dither volume* provides great flexibility in dither cell design, by decoupling different intensity levels. General dither volumes can be large - a 64×64×256 dither volume, for example, has a size of 128KB. They are also inefficient to access since each color component requires the retrieval of a different bit from the volume. In practice, there is no need to fully decouple each layer of the dither volume. Each dot column of the volume can be implemented as a fixed set of thresholds rather than 256 separate bits. Using three 8-bit thresholds, for example, only consumes 24 bits. Now, n thresholds define n+1 intensity intervals, within which the corresponding dither cell location is alternately not set or set. The contone pixel value being dithered uniquely selects one of the n+1 intervals, and this determines the value of the corresponding output dot.

We dither the contone data using a triple-threshold 64×64×3×8-bit (12KB) dither volume. The three thresholds form a convenient 24-bit value which can be retrieved from the dither cell ROM in one cycle. If dither cell registration is desired between color planes, then the same triple-threshold value can be retrieved once and used to dither each color component. If dither cell registration is not desired, then the dither cell can be split into four subcells and stored in four separately addressable ROMs from which four different triple-threshold values can be retrieved in parallel in one cycle. Using the addressing scheme shown below, the four color planes share the same dither cell at vertical and/or horizontal offsets of 32 dots from each other.

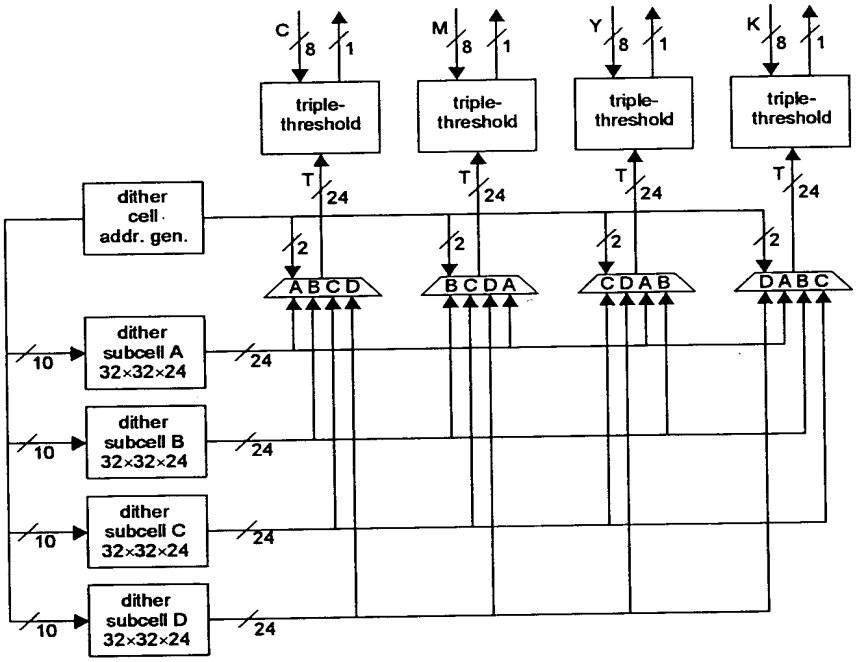


Figure 33. Multi-threshold dither

The triple-threshold unit converts a triple-threshold value and an intensity value into an interval and thence a one or zero bit. The triple-thresholding rules are shown in Table 24. The corresponding logic is shown in Figure 34.

Table 24. Triple-thresholding rules

Interval	output
$V < T_1$	0
$T_1 < V \leq T_2$	1
$T_2 < V \leq T_3$	0
$T_3 < V$	1

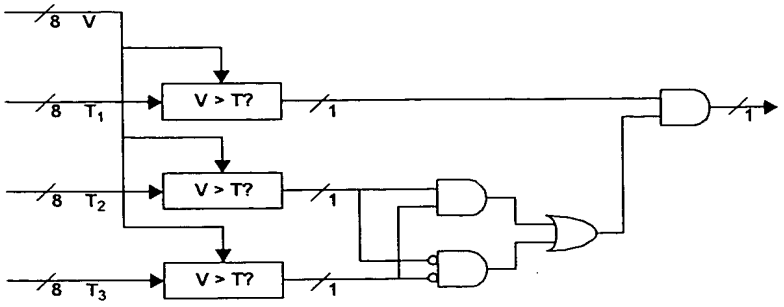


Figure 34. Triple-threshold

7.2.4.2 Composite

The composite unit composites a black layer dot over a halftoned CMYK layer dot. If the black layer opacity is one, then the halftoned CMY is set to zero.

Given a 4-bit halftoned color $C_cM_cY_cK_c$ and a 1-bit black layer opacity K_b, the composite and clip logic is as defined in Table 25.

Table 25. Composite logic

color channel	condition
C	$C_c \wedge \neg K_b$
M	$M_c \wedge \neg K_b$
Y	$Y_c \wedge \neg K_b$
K	$K_c \vee K_b$

7.2.4.3 Clock Enable Generator

The clock enable generator generates enable signals for clocking the contone CMYK pixel input, the black dot input, and the CMYK dot output.

As described earlier, the contone pixel input buffer is used as both a line buffer and a FIFO. Each line is read once and then used *contone scale factor* times. FIFO write address wrapping is disabled until the start of the final replicated use of the line, at which time the clock enable generator generates a *contone line advance enable* signal which enables wrapping.

The clock enable generator also generates an *even* signal which is used to select the even or odd set of output dot FIFOs, and a *margin* signal which is used to generate white dots when the current dot position is in the left or right margin of the page.

The clock enable generator uses a set of counters. The internal logic of the counters is defined in Table 26. The logic of the clock enable signals is defined in Table 27.

Table 26. Clock enable generator counter logic

counter	abbr.	w.	data	load condition	decrement condition
dot	D	14	page width	RP[a] ∨ EOL[b]	(D>0) ∧ clk
line	L	15	page length	RP	(L>0) ∧ EOL
left margin	LM	14	left margin	RP ∨ EOL	(LM>0) ∧ clk
right margin	RM	14	right margin	RP ∨ EOL	(RM>0) ∧ clk
even/odd dot	E	1	0	RP ∨ EOL	clk
black dot	BD	14	black width	RP ∨ EOL	(LM=0) ∧ (BD>0) ∧ clk
contone dot	CD	14	contone width	RP ∨ EOL	(LM=0) ∧ (CD>0) ∧ clk
contone sub-pixel	CSP	4	contone scale factor	RP ∨ EOL ∨ (CSP=0)	(LM=0) ∧ clk
contone sub-line	CSL	4	contone scale factor	RP ∨ (CSL=0)	EOL ∧ clk

a. RP (reset page) condition: external signal
b. EOL (end-of-line) condition: (D=0) ∧ (BD=0) ∧ (CD=0)

Table 27. Clock enable generator output signal logic

output signal	condition
output dot clock enable	(D>0) ∧ ¬EOP[a]
black dot clock enable	(LM=0) ∧ (BD>0) ∧ ¬EOP
contone pixel clock enable	(LM=0) ∧ (CD>0) ∧ (CSP=0) ∧ ¬EOP
contone line advance enable	(CSL=0) ∧ ¬EOP
even	E=0
margin	(LM=0) ∨ (RM=0)

a. EOP (end-of-page) condition: L=0

7.2.4.4 Timing

The HCU has an output rate of 124M 4-bit CMYK pixels/s. Since it generates one pixel per clock, it must be clocked at at least 124MHz.

7.3 PRINTHEAD INTERFACE

CePrint uses an 8.5" CMYK Memjet printhead, as described in Section 9. The printhead consists of 17 segments arranged in 2 segment groups. The first segment group contains 9 segments, and the second group contains 8 segments. There are 13,600 nozzles of each color in the printhead, making a total of 54,400 nozzles.

The printhead interface is a standard Memjet printhead interface, as described in Section 10, configured with the following operating parameters:

- MaxColors = 4
- SegmentsPerXfer = 9
- SegmentGroups = 2

Although the printhead interface has a number of external connections, not all used for an 8.5" printhead, so not all are connected to external pins on the CCP. Specifically, the value for SegmentGroups implies that there are only 2 SRClock pins and 2 SenseSegSelect pins. All 36 ColorData pins, however, are required.

7.3.1 Timing

CePrint prints an 8.3" × 11.7" page in 2 seconds. The printhead must therefore print 18,720 lines (11.7" × 1600 dpi) in 2 seconds, which yields a line time of about 107µs. Within the printhead interface, a single Print Cycle and a single Load Cycle must both complete within this time. In addition, the paper must advance by about 16µm in the same time.

In high-speed print mode the Memjet printhead can print an entire line in 100µs. Since all segments fire at the same time 544 nozzles are fired simultaneously with each firing pulse. This leaves 7µs for other tasks between each line.

The 1600 SRClock pulses (800 each of SRClock1 and SRClock2) to the printhead (SRClock1 has 36 bits of valid data, and SRClock2 has 32 bits of valid data) must also take place within the 107µs line time. Restricting the timing to 100µs, the length of an SRClock pulse cannot exceed 100µs / 1600 = 62.5ns. The printhead must therefore be clocked at 16MHz.

The printhead interface has a nominal pixel rate of 124M 4-bit CMYK pixels/s. However, because it is only active for 100µs out of every 107µs, it must be clocked at at least 140MHz. This can be increased to 144MHz to make it an integer multiple of the printhead speed.

7.4 PROCESSOR AND MEMORY

7.4.1 Processor

The processor runs the control program which synchronizes the other functional units during page reception, expansion and printing. It also runs the device drivers for the various external interfaces, and responds to user actions through the user interface.

It must have low interrupt latency, to provide efficient DMA management, but otherwise does not need to be particularly high-performance.

7.4.2 DMA Controller

The DMA controller supports single-address transfers on 29 channels (see Table 28). It generates vectored interrupts to the processor on transfer completion.

Table 28. DMA channel usage

functional unit	input channels	output channels
host interface	-	1
inter-CCP interface	1	1
EDRL expander	1	1
JPEG decoder	1	8

Table 28. DMA channel usage

functional unit	input channels	output channels
halftoner/compositor	2	8
speaker interface	1	-
printhead interface	4	-
	10	19
		29

7.4.3 Program ROM

The program ROM holds the CCP control program which is loaded into main memory during system boot.

7.4.4 Rambus Interface

The Rambus interface provides the high-speed interface to the external 8MB (64Mbit) Rambus DRAM (RDRAM).

7.5 EXTERNAL INTERFACES

7.5.1 Host Interface

The host interface provides a connection to the host processor with a speed of at least 1.5MB/s (or 3MB/s for the double-sided version of CePrint).

7.5.2 Speaker Interface

The speaker interface (Figure 35) contains a small FIFO used for DMA-mediated transfers of sound clips from main memory, an 8-bit digital-to-analog converter (DAC) which converts each 8-bit sample value to a voltage, and an amplifier which feeds the external speaker. When the FIFO is empty it outputs a zero value.

The speaker interface is clocked at the frequency of the sound clips.

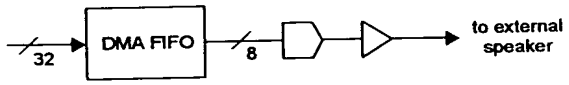


Figure 35. Speaker interface

The processor outputs a sound clip to the speaker simply by programming the DMA channel of the speaker interface.

7.5.3 Parallel Interface

The parallel interface provides I/O on a number of parallel external signal lines.

It allows the processor to sense or control the devices listed in Table 29.

Table 29. Parallel Interface devices

parallel interface devices
power button
power LED
out-of-paper LED
out-of-ink LED
media sensor
paper pick-up roller position sensor
paper tray drive position sensor
paper pick-up motor
paper tray ejector motor
transfer roller stepper motor

7.5.4 Serial Interface

The serial interface provides two standard low-speed serial ports.

One port is used to connect to the master QA chip. The other is used to connect to the QA chip in the ink cartridge. The processor-mediated protocol between the two is used to authenticate the ink cartridge [12,13]. The processor can then retrieve ink characteristics from the QA chip, as well as the remaining volume of each ink. The processor uses the ink characteristics to properly configure the Memjet printhead. It uses the remaining ink volumes, updated on a page-by-page basis with ink consumption information accumulated by the printhead interface, to ensure that it never allows the printhead to be damaged by running dry.

7.5.4.1 Ink Cartridge QA Chip

The QA chip in the ink cartridge contains information required for maintaining the best possible print quality, and is implemented using an authentication chip [12,13]. The 256 bits of data in the authentication chip are allocated as follows:

Table 30. Ink cartridge's 256 bits (16 entries of 16-bits)

M[n]	access	width	description
0	RO[a]	16	Basic header, flags etc.
1	RO	16	Serial number.
2	RO	16	Batch number.
3	RO	16	Reserved for future expansion. Must be 0.
4	RO	16	Cyan ink properties.
5	RO	16	Magenta ink properties.
6	RO	16	Yellow ink properties.
7	RO	16	Black ink properties.
8-9	DO[b]	32	Cyan ink remaining, in nanolitres.
10-11	DO	32	Magenta ink remaining, in nanolitres.
12-13	DO	32	Yellow ink remaining, in nanolitres.
14-15	DO	32	Black ink remaining, in nanolitres.

a. read only (RO)
b. decrement only (DO)

Before each page is printed, the processor must check the amount of ink remaining to ensure there is enough for an entire worst-case page. Once the page has been printed, the processor multiplies the total number of drops of each color (obtained from the printhead interface) by the drop volume. The amount of printed ink is subtracted from the amount of ink remaining. The unit of measurement for ink remaining is nanolitres, so 32 bits can represent over 4 liters of ink. The amount of ink used for a page must be rounded up to the nearest nanolitre (i.e. approximately 1000 printed dots).

7.5.5 Inter-CCP Interface

The inter-CCP interface provides a bi-directional high-speed serial communications link to a second CCP, and is used in multi-CCP configurations such as the double-sided version of the printer which contains two CCPs.

The link has a minimum speed of 30MB/s, to support timely distribution of page data, and may be implemented using a technology such as IEEE 1394 or Rambus.

7.5.6 JTAG Interface

A standard JTAG (Joint Test Action Group) interface is included for testing purposes. Due to the complexity of the chip, a variety of testing techniques are required, including BIST (Built In Self Test) and functional block isolation. An overhead of 10% in chip area is assumed for overall chip testing circuitry.

8 Double-Sided Printing

The double-sided version of CePrint contains two complete printing units - one for the front of the paper, one for the back. Each printing unit consists of a printer controller, a Memjet printhead, and a transfer roller. Both printing units share the same ink supply.

The back side printing unit acts as the master unit. It is responsible for global printer functions, such as communicating with the host, handling the ink cartridge, handling the user interface, and controlling the paper transport. The front side printing unit acts as a slave unit. It obtains pages from the host processor via the master unit, and is synchronized by the master unit during printing.

Both printer controllers consist of a CePrint central processor (CCP) and a local 8MB RDRAM. The external interfaces of the master unit are used in the same way as in the single-sided version of CePrint, but only the memory interface and the printhead interface of the slave unit are used. An external master/slave pin on the CCP selects the mode of operation.

This dual printer controller configuration is illustrated in Figure 36.

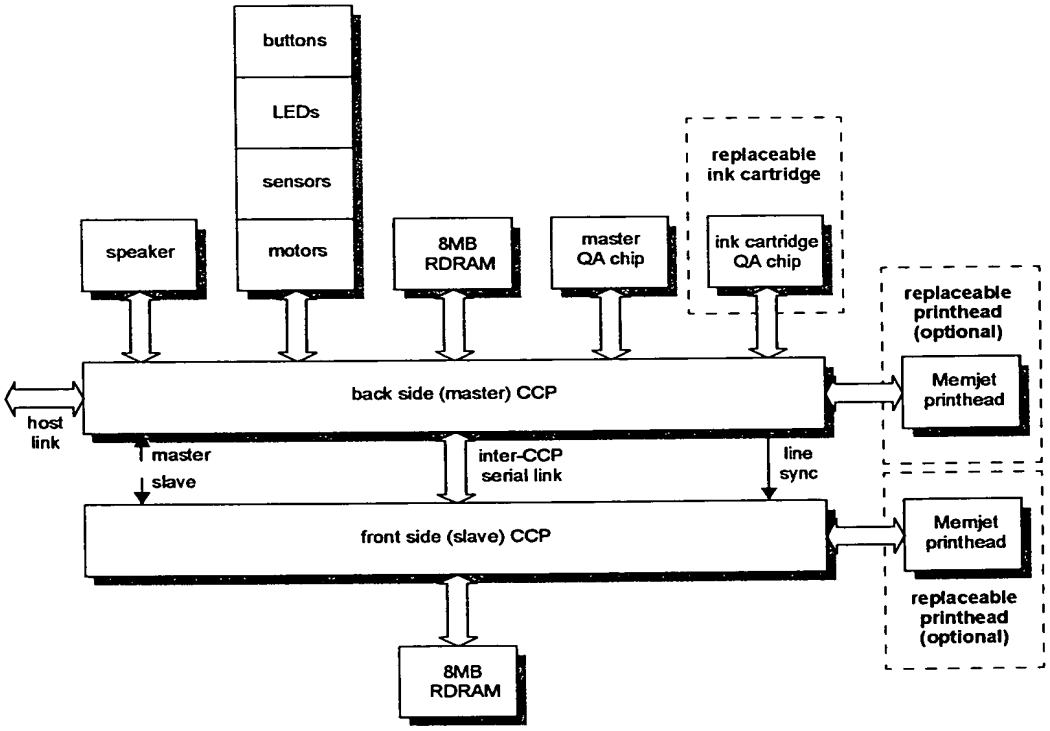


Figure 36. Dual printer controller configuration

8.1 Page Delivery and Distribution

The master CCP presents a unified view of the printer to the host processor. It hides the presence of the slave CCP.

Pages are transmitted from the host processor to the printer in page order. The first page of a document is always a front side page, and front side and back side pages are always interleaved. Thus odd-numbered pages are front side pages, and even-numbered pages are back side pages. To print in single-sided mode on either the front side or back side of the paper, the host must send appropriate blank pages to the printer. The printer expects a page description for every page.

When the master CCP receives a page command from the host processor relating to an odd-numbered page it routes the command to the slave CCP via the inter-CCP serial link. To avoid imposing undue restrictions on the host link and its protocol, each command is received in its entirety and stored in the master's local memory before being forwarded to the slave. This introduces only a small delay because the inter-CCP link is fast. To ensure that the master CCP always has a page buffer available for a page destined for the slave, the master is deliberately made the back side CCP, so that it receives the front side odd-numbered page before it receives the matching back side even-numbered page.

8.2 SYNCHRONIZED PRINTING

Once the master CCP and the slave CCP have received their pages, the master CCP initiates actual printing. This consists of starting the page expansion and printing processes in the master CCP, and initiating the same processes in the slave CCP via a command sent over the inter-CCP serial link.

The achieve perfect registration between the front side and back side printed pages, the printhead interfaces of both CCPs are synchronized to a common line synchronization signal. The synchronization signal is generated by the master CCP.

Once the printing pipelines in both CCPs are sufficiently primed, as indicated by the stall status of the line loader/format unit (LLFU) of the printhead interface (Section 10.4), the master CCP starts the line synchronization generator unit (LSGU) of the printhead interface (Section 10.2). The master CCP obtains the status of the slave CCP LLFU via a poll sent over the inter-CCP serial link.

After the printing of a page, or more frequently, the master obtains ink consumption information from the slave over the inter-CCP link. It uses this to update the remaining ink volume in the ink cartridge, as described in Section 7.5.4.1.

The master and slave CCPs also exchange error events and host-initiated printer reset commands over the inter-CCP link.

9 Memjet Printhead

A Memjet printhead is a drop-on-demand 1600 dpi inkjet printer that produces bi-level dots in up to 4 colors to produce a printed page of a particular width. Since the printhead prints dots at 1600 dpi, each dot is approximately 22.5µm in diameter, and spaced 15.875µm apart. Because the printing is bi-level, the input image should be dithered or error-diffused for best results.

Typically a Memjet printhead for a particular application is page-width. This enables the printhead to be stationary and allows the paper to move past the printhead. Figure 37 illustrates a typical configuration.

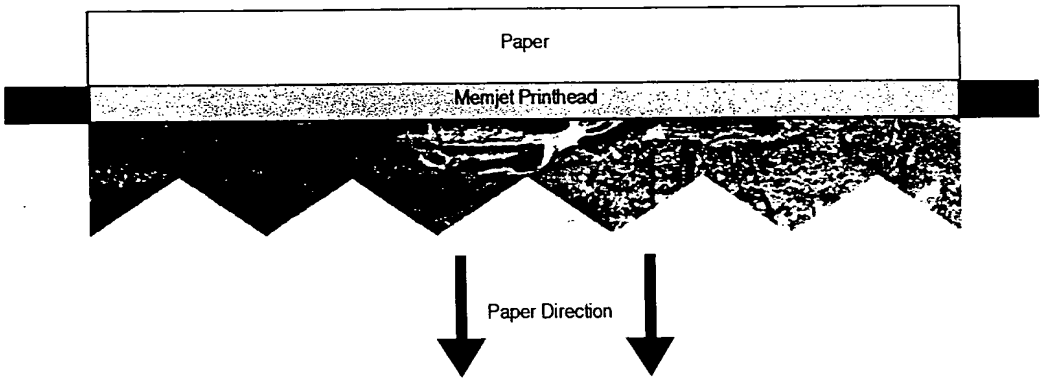


Figure 37. A Memjet printhead

A Memjet printhead is composed of a number of identical 1/2 inch Memjet segments. The segment is therefore the basic building block for constructing a printhead.

9.1 THE STRUCTURE OF A MEMJET SEGMENT

This section examines the structure of a single segment, the basic building block for constructing Memjet printheads.

9.1.1 Grouping of Nozzles Within a Segment

The nozzles within a single segment are grouped for reasons of physical stability as well as minimization of power consumption during printing. In terms of physical stability, a total of 10 nozzles share the same ink reservoir. In terms of power consumption, groupings are made to enable a low-speed and a high-speed printing mode.

Memjet segments support two printing speeds to allow speed/power consumption tradeoffs to be made in different product configurations.

In the low-speed printing mode, 4 nozzles of each color are fired from the segment at a time. The exact number of nozzles fired depends on how many colors are present in the printhead. In a four color (e.g. CMYK) printing environment this equates to 16 nozzles firing simultaneously. In a three color (e.g. CMY) printing environment this equates to 12 nozzles firing simultaneously. To fire all the nozzles in a segment, 200 different sets of nozzles must be fired.

In the high-speed printing mode, 8 nozzles of each color are fired from the segment at a time. The exact number of nozzles fired depends on how many colors are present in the printhead. In a four color (e.g. CMYK) printing environment this equates to 32 nozzles firing simultaneously. In a three color (e.g. CMY) printing environment this equates to 24 nozzles firing simultaneously. To fire all the nozzles in a segment, 100 different sets of nozzles must be fired.

The power consumption in the low-speed mode is half that of the high-speed mode. Note, however, that the energy consumed to print a page is the same in both cases.

9.1.1.1 10 Nozzles Make a Pod

A single pod consists of 10 nozzles sharing a common ink reservoir. 5 nozzles are in one row, and 5 are in another. Each nozzle produces dots 22.5µm in diameter spaced on a 15.875µm grid to print at 1600 dpi. Figure 38 shows the arrangement of a single pod, with the nozzles numbered according to the order in which they must be fired.

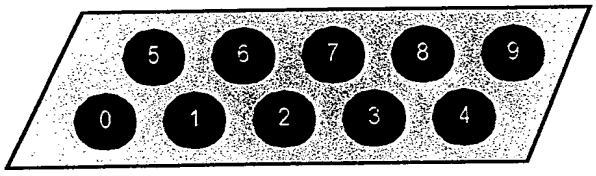


Figure 38. A single pod, numbered by firing order

Although the nozzles are fired in this order, the relationship of nozzles and physical placement of dots on the printed page is different. The nozzles from one row represent the even dots from one line on the page, and the nozzles on the other row represent the odd dots from the adjacent line on the page. Figure 39 shows the same pod with the nozzles numbered according to the order in which they must be loaded.

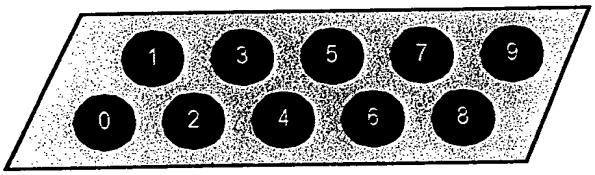


Figure 39. A single pod, numbered by load order

The nozzles within a pod are therefore logically separated by the width of 1 dot. The exact distance between the nozzles will depend on the properties of the Memjet firing mechanism. The printhead is designed with staggered nozzles designed to match the flow of paper.

9.1.1.2 1 Pod of Each Color Makes a Chromapod

One pod of each color are grouped together into a ***chromapod***. The number of pods in a chromapod will depend on the particular application. In a monochrome printing system (such as one that prints only black), there is only a single color and hence a single pod. Photo printing application printheads require 3 colors (cyan, magenta, yellow), so Memjet segments used for these applications will have 3 pods per chromapod (one pod of each color). The expected maximum number of pods in a chromapod is 4, as used in a CMYK (cyan, magenta, yellow, black) printing system (such as a desktop printer). This maximum of 4 colors is not imposed by any physical constraints - it is merely an expected maximum

from the expected applications (of course, as the number of colors increases the cost of the segment increases and the number of these larger segments that can be produced from a single silicon wafer decreases).

A chromapod represents different color components of the same horizontal set of 10 dots on different lines. The exact distance between different color pods depends on the Memjet operating parameters, and may vary from one Memjet design to another. The distance is considered to be a constant number of dot-widths, and must therefore be taken into account when printing: the dots printed by the cyan nozzles will be for different lines than those printed by the magenta, yellow or black nozzles. The printing algorithm must allow for a variable distance up to about 8 dot-widths between colors. Figure 40 illustrates a single chromapod for a CMYK printing application.

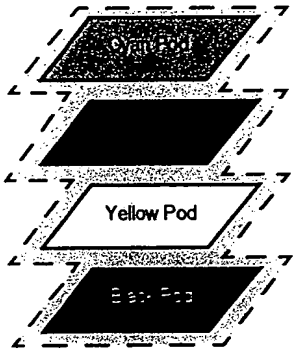


Figure 40. A Single Chromapod Contains 1 Pod of each Color

9.1.1.3 5 Chromapods Make a Podgroup

5 chromapods are organized into a single ***podgroup***. A podgroup therefore contains 50 nozzles for each color. The arrangement is shown in Figure 41, with chromapods numbered 0-4 and using a CMYK chromapod as the example. Note that the distance between adjacent chromapods is exaggerated for clarity.

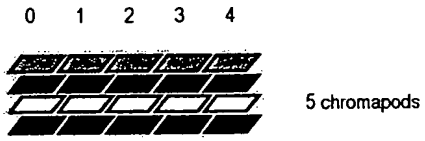


Figure 41. A Single Podgroup Contains 5 Chromapods

9.1.1.4 2 Podgroups Make a Phasegroup

2 podgroups are organized into a single ***phasegroup***. The phasegroup is so named because groups of nozzles within a phasegroup are fired simultaneously during a given firing phase (this is explained in more detail below). The formation of a phasegroup from 2 podgroups is entirely for the purposes of low-speed and high-speed printing via 2 PodgroupEnable lines.

During low-speed printing, only one of the two PodgroupEnable lines is set in a given firing pulse, so only one podgroup of the two fires nozzles. During high-speed printing, both PodgroupEnable lines are set, so both podgroups fire nozzles. Consequently a low-speed print takes twice as long as a high-speed print, since the high-speed print fires twice as many nozzles at once.

Figure 42 illustrates the composition of a phasegroup. The distance between adjacent podgroups is exaggerated for clarity.

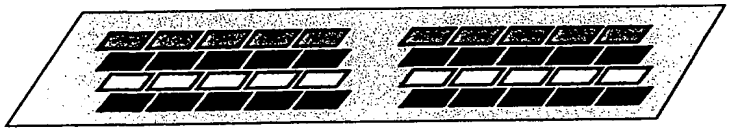


Figure 42. A single phasegroup contains 2 podgroups

9.1.1.5 2 Phasegroups Make a Firegroup

Two phasegroups (PhasegroupA and PhasegroupB) are organized into a single ***firegroup***, with 4 firegroups in each segment. Firegroups are so named because they all fire the same nozzles simultaneously. Two enable lines, AEnable and BEnable, allow the firing of PhasegroupA nozzles and PhasegroupB nozzles independently as different firing phases. The arrangement is shown in Figure 43. The distance between adjacent groupings is exaggerated for clarity.

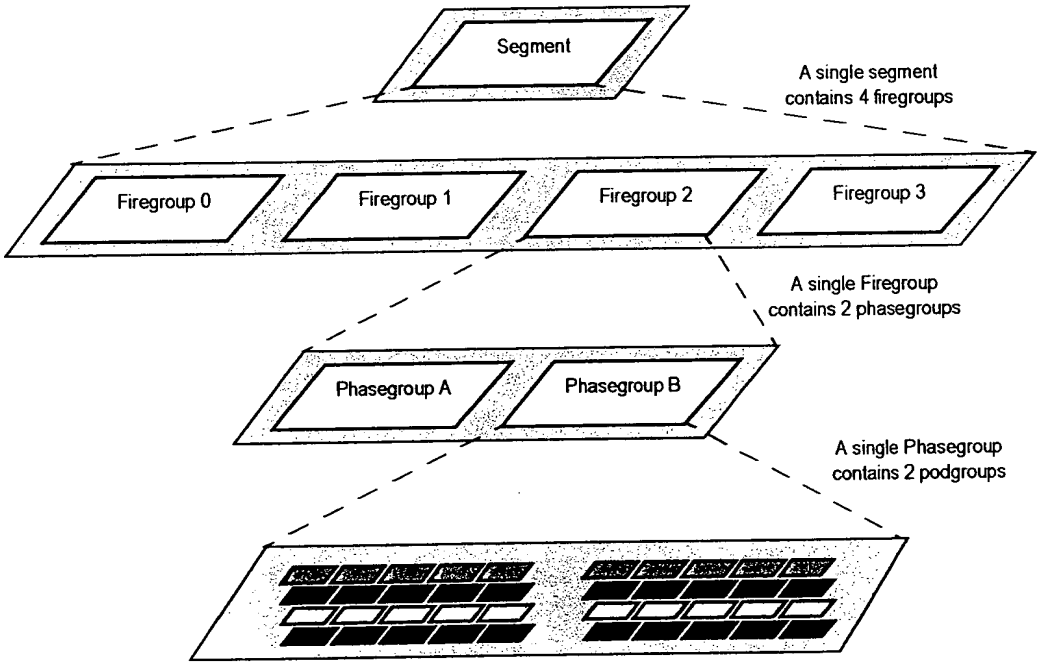


Figure 43. Relationship Between Segments, Firegroups, Phasegroups, Podgroups and ChromaPods

9.1.1.6 Nozzle Grouping Summary

Table 31 is a summary of the nozzle groupings in a segment assuming a CMYK chromapod.

Table 31. Nozzle Groupings for a single segment

Name of Grouping	Composition	Replication Ratio	Nozzle Count
Nozzle	Base unit	1:1	1
Pod	Nozzles per pod	10:1	10
Chromapod	Pods per chromapod	C:1	10C
Podgroup	Chromapods per podgroup	5:1	50C
Phasegroup	Podgroups per phasegroup	2:1	100C
Firegroup	Phasegroups per firegroup	2:1	200C
Segment	Firegroups per segment	4:1	800C

The value of C, the number of colors contained in the segment, determines the total number of nozzles.

- With a 4 color segment, such as CMYK, the number of nozzles per segment is 3,200.
- With a 3 color segment, such as CMY, the number of nozzles per segment is 2,400.
- In a monochrome environment, the number of nozzles per segment is 800.

9.1.2 Load and Print Cycles

A single segment contains a total of 800C nozzles, where C is the number of colors in the segment. A *Print Cycle* involves the firing of up to all of these nozzles, dependent on the information to be printed. A *Load Cycle* involves the loading up of the segment with the information to be printed during the subsequent Print Cycle.

Each nozzle has an associated ***NozzleEnable*** bit that determines whether or not the nozzle will fire during the Print Cycle. The NozzleEnable bits (one per nozzle) are loaded via a set of shift registers.

Logically there are C shift registers per segment (one per color), each 800 deep. As bits are shifted into the shift register for a given color they are directed to the lower and upper nozzles on alternate pulses. Internally, each 800-deep shift register is comprised of two 400-deep shift registers: one for the upper nozzles, and one for the lower nozzles. Alternate bits are shifted into the alternate internal registers. As far as the external interface is concerned however, there is a single 800 deep shift register.

Once all the shift registers have been fully loaded (800 load pulses), all of the bits are transferred in parallel to the appropriate NozzleEnable bits. This equates to a single parallel transfer of 800C bits. Once the transfer has taken place, the Print Cycle can begin. The Print Cycle and the Load Cycle can occur simultaneously as long as the parallel load of all NozzleEnable bits occurs at the end of the Print Cycle.

9.1.2.1 Load Cycle

The Load Cycle is concerned with loading the segment's shift registers with the next Print Cycle's NozzleEnable bits.

Each segment has C inputs directly related to the C shift registers (where C is the number of colors in the segment). These inputs are named ***ColorNData***, where N is 1 to C (for example, a 4 color segment would have 4 inputs labeled ***Color1Data***, ***Color2Data***, ***Color3Data*** and ***Color4Data***). A single pulse on the ***SRClock*** line transfers C bits into the appropriate shift registers. Alternate pulses transfer bits to the lower and upper nozzles respectively. A total of 800 pulses are required for the complete transfer of data. Once all 800C bits have been transferred, a single pulse on the ***PTransfer*** line causes the parallel transfer of data from the shift registers to the appropriate NozzleEnable bits.

The parallel transfer via a pulse on PTransfer must take place *after* the Print Cycle has finished. Otherwise the NozzleEnable bits for the line being printed will be incorrect.

It is important to note that the odd and even dot outputs, although printed during the same Print Cycle, do not appear on the same physical output line. The physical separation of odd and even nozzles within the printhead, as well as separation between nozzles of different colors ensures that they will produce dots on different lines of the page. This relative difference must be accounted for when loading the data into the printhead. The actual difference in lines depends on the characteristics of the inkjet mechanism used in the printhead. The differences can be defined by variables D_1 and D_2 where D_1 is the distance between nozzles of different colors, and D_2 is the distance between nozzles of the same color. Table 32 shows the dots transferred to a C color segment on the first 4 pulses.

Table 32. Order of Dots Transferred to a Segment

Pulse	Dot	Color1 Line	Color2 Line	Color3 Line	ColorC Line
1	0	N	$N+D_1$[a]	$N+2D_1$	$N+(C-1)D_1$
2	1	$N+D_2$[b]	$N+D_1+D_2$	$N+2D_1+D_2$	$N+(C-1)D_1+D_2$
3	2	N	$N+D_1$	$N+2D_1$	$N+(C-1)D_1$
4	3	$N+D_2$	$N+D_1+D_2$	$N+2D_1+D_2$	$N+(C-1)D_1+D_2$

a. D_1 = number of lines between the nozzles of one color and the next (likely = 4-8)

b. D_2 = number of lines between two rows of nozzles of the same color (likely = 1)

And so on for all 800 pulses.

Data can be clocked into a segment at a maximum rate of 20 MHz, which will load the entire 800C bits of data in 40µs.

9.1.2.2 Print Cycle

A single Memjet printhead segment contains 800 nozzles. To fire them all at once would consume too much power and be problematic in terms of ink refill and nozzle interference. This problem is made more apparent when we consider that a Memjet printhead is composed of multiple 1/2 inch segments, each with 800 nozzles. Consequently two firing modes are defined: a low-speed printing mode and a high-speed printing mode:

- In the low-speed print mode, there are 200 phases, with each phase firing 4C nozzles (C per firegroup, where C is the number of colors).
- In the high-speed print mode, there are 100 phases, with each phase firing 8C nozzles, (2C per firegroup, where C is the number of colors).

The nozzles to be fired in a given firing pulse are determined by

- 3 bits ***ChromapodSelect*** (select 1 of 5 chromapods from a firegroup)
- 4 bits ***NozzleSelect*** (select 1 of 10 nozzles from a pod)
- 2 bits of ***PodgroupEnable*** lines (select 0, 1, or 2 podgroups to fire)

When one of the PodgroupEnable lines is set, only the specified Podgroup's 4 nozzles will fire as determined by ChromapodSelect and NozzleSelect. When both of the PodgroupEnable lines are set, both of the podgroups will fire their nozzles. For the low-speed mode, two fire pulses are required, with PodgroupEnable = 10 and 01 respectively. For the high-speed mode, only one fire pulse is required, with PodgroupEnable = 11.

The duration of the firing pulse is given by the ***AEnable*** and ***BEnable*** lines, which fire the PhasegroupA and PhasegroupB nozzles from all firegroups respectively. The typical duration of a firing pulse is 1.3 - 1.8 µs. The duration of a pulse depends on the viscosity of the ink (dependent on temperature and ink characteristics) and the amount of power available to the printhead. See Section 9.1.3 on page 70 for details on feedback from the printhead in order to compensate for temperature change.

The AEnable and BEnable are separate lines in order that the firing pulses can overlap. Thus the 200 phases of a low-speed Print Cycle consist of 100 A phases and 100 B phases, effectively giving 100 sets of Phase A and Phase B. Likewise, the 100 phases of a high-speed print cycle consist of 50 A phases and 50 B phases, effectively giving 50 phases of phase A and phase B.

Figure 44 shows the AEnable and BEnable lines during a typical Print Cycle. In a high-speed print there are 50 2µs cycles, while in a low-speed print there are 100 2µs cycles.

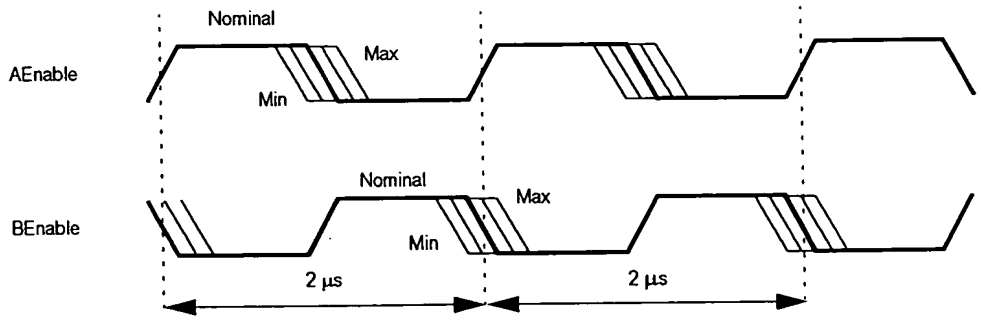


Figure 44. AEnable and BEnable During a Typical Print Cycle

For the high-speed printing mode, the firing order is:

- ChromapodSelect 0, NozzleSelect 0, PodgroupEnable 11 (Phases A and B)
- ChromapodSelect 1, NozzleSelect 0, PodgroupEnable 11 (Phases A and B)
- ChromapodSelect 2, NozzleSelect 0, PodgroupEnable 11 (Phases A and B)
- ChromapodSelect 3, NozzleSelect 0, PodgroupEnable 11 (Phases A and B)
- ChromapodSelect 4, NozzleSelect 0, PodgroupEnable 11 (Phases A and B)
- ChromapodSelect 0, NozzleSelect 1, PodgroupEnable 11 (Phases A and B)
- ...
- ChromapodSelect 3, NozzleSelect 9, PodgroupEnable 11 (Phases A and B)
- ChromapodSelect 4, NozzleSelect 9, PodgroupEnable 11 (Phases A and B)

For the low-speed printing mode, the firing order is similar. For each phase of the high speed mode where PodgroupEnable was 11, two phases of PodgroupEnable = 01 and 10 are substituted as follows:

- ChromapodSelect 0, NozzleSelect 0, PodgroupEnable 01 (Phases A and B)
- ChromapodSelect 0, NozzleSelect 0, PodgroupEnable 10 (Phases A and B)
- ChromapodSelect 1, NozzleSelect 0, PodgroupEnable 01 (Phases A and B)
- ChromapodSelect 1, NozzleSelect 0, PodgroupEnable 10 (Phases A and B)
- ...
- ChromapodSelect 3, NozzleSelect 9, PodgroupEnable 01 (Phases A and B)
- ChromapodSelect 3, NozzleSelect 9, PodgroupEnable 10 (Phases A and B)
- ChromapodSelect 4, NozzleSelect 9, PodgroupEnable 01 (Phases A and B)
- ChromapodSelect 4, NozzleSelect 9, PodgroupEnable 10 (Phases A and B)

When a nozzle fires, it takes approximately 100μs to refill. The nozzle cannot be fired before this refill time has elapsed. This limits the fastest printing speed to 100μs per line. In the high-speed print mode, the time to print a line is 100μs, so the time between firing a nozzle from one line to the next matches the refill time. The low-speed print mode is slower than this, so is also acceptable.

The firing of a nozzle also causes acoustic perturbations for a limited time within the common ink reservoir of that nozzle's pod. The perturbations can interfere with the firing of another nozzle within the same pod. Consequently, the firing of nozzles within a pod should be offset from each other as long as possible. We therefore fire four nozzles from a chromapod (one nozzle per color) and then move onto the next chromapod within the podgroup.

- In the low-speed printing mode the podgroups are fired separately. Thus the 5 chromapods within both podgroups must all fire before the first chromapod fires again, totalling 10 × 2μs cycles. Consequently each pod is fired once per 20μs.
- In the high-speed printing mode, the podgroups are fired together. Thus the 5 chromapods within a single podgroups must all fire before the first chromapod fires again, totalling 5 × 2μs cycles. Consequently each pod is fired once per 10 μs.

As the ink channel is 300μm long and the velocity of sound in the ink is around 1500m/s, the resonant frequency of the ink channel is 2.5MHz. Thus the low-speed mode allows 50 resonant cycles for the acoustic pulse to dampen, and the high-speed mode allows 25 resonant cycles. Consequently any acoustic interference is minimal in both cases.

9.1.3 Feedback from a Segment

A segment produces several lines of feedback. The feedback lines are used to adjust the timing of the firing pulses. Since multiple segments are collected together into a printhead, it is effective to share the feedback lines as a tri-state bus, with only one of the segments placing the feedback information on the feedback lines.

A pulse on the segment's ***SenseSegSelect*** line ANDed with data on ***Color1Data*** selects if the particular segment will provide the feedback. The feedback sense lines will come from that segment until the next SenseSegSelect pulse. The feedback sense lines are as follows:

- ***Tsense*** informs the controller how hot the printhead is. This allows the controller to adjust timing of firing pulses, since temperature affects the viscosity of the ink.
- ***Vsense*** informs the controller how much voltage is available to the actuator. This allows the controller to compensate for a flat battery or high voltage source by adjusting the pulse width.
- ***Rsense*** informs the controller of the resistivity (Ohms per square) of the actuator heater. This allows the controller to adjust the pulse widths to maintain a constant energy irrespective of the heater resistivity.
- ***Wsense*** informs the controller of the width of the critical part of the heater, which may vary up to ± 5% due to lithographic and etching variations. This allows the controller to adjust the pulse width appropriately.

9.1.4 Preheat Cycle

The printing process has a strong tendency to stay at the equilibrium temperature. To ensure that the first section of the printed photograph has a consistent dot size, the equilibrium temperature must be met *before* printing any dots. This is accomplished via a preheat cycle.

The Preheat cycle involves a single Load Cycle to all nozzles of a segment with 1s (i.e. setting all nozzles to fire), and a number of short firing pulses to each nozzle. The duration of the pulse must be insufficient to fire the drops, but enough to heat up the ink. Altogether about 200 pulses for each nozzle are required, cycling through in the same sequence as a standard Print Cycle.

Feedback during the Preheat mode is provided by Tsense, and continues until equilibrium temperature is reached (about 30° C above ambient). The duration of the Preheat mode is around 50 milliseconds, and depends on the ink composition.

Preheat is performed before each print job. This does not affect performance as it is done while the data is being transferred to the printer.

9.1.5 Cleaning Cycle

In order to reduce the chances of nozzles becoming clogged, a cleaning cycle can be undertaken before each print job. Each nozzle is fired a number of times into an absorbent sponge.

The cleaning cycle involves a single Load Cycle to all nozzles of a segment with 1s (i.e. setting all nozzles to fire), and a number of firing pulses to each nozzle. The nozzles are cleaned via the same nozzle firing sequence as a standard Print Cycle. The number of times that each nozzle is fired depends upon the ink composition and the time that the

printer has ben idle. As with preheat, the cleaning cycle has no effect on printer performance.

9.1.6 Printhead Interface Summary

Each segment has the following connections to the bond pads:

Table 33. Segment Interface Connections

Name	Lines	Description
Chromapod Select	3	Select which chromapod will fire (0-4)
NozzleSelect	4	Select which nozzle from the pod will fire (0-9)
PodgroupEnable	2	Enable the podgroups to fire (choice of: 01, 10, 11)
AEnable	1	Firing pulse for podgroup A
BEnable	1	Firing pulse for podgroup B
ColorNData	C	Input to shift registers (1 bit for each of C colors in the segment)
SRClock	1	A pulse on SRClock (ShiftRegisterClock) loads C bits from ColorData into the C shift registers.
PTransfer	1	Parallel transfer of data from the shift registers to the internal NozzleEnable bits (one per nozzle).
SenseSegSelect	1	A pulse on SenseSegSelect ANDed with data on Color1Data selects the sense lines for this segment.
Tsense	1	Temperature sense
Vsense	1	Voltage sense
Rsense	1	Resistivity sense
Wsense	1	Width sense
Logic GND	1	Logic ground
Logic PWR	1	Logic power
V-	21	Actuator Ground
V+	21	Actuator Power
TOTAL	62+C	(if C is 4, Total = 66)

9.2 MAKING MEMJET PRINTHEADS OUT OF SEGMENTS

A Memjet printhead is composed of a number of identical 1/2 inch printhead segments. These 1/2 inch segments are manufactured together or placed together after manufacture to produce a printhead of the desired length. Each 1/2 inch segments prints 800 1600 dpi bi-level dots in up to 4 colors over a different part of the page to produce the final image. Although each segment produces 800 dots of the final image, each dot is represented by a combination of colored inks.

A 4-inch printhead, for example, consists of 8 segments, typically manufactured as a monolithic printhead. In a typical 4-color printing application (cyan, magenta, yellow, black), each of the segments prints bi-level cyan, magenta, yellow and black dots over a different part of the page to produce the final image. The positions of the segments are shown in Figure 45.

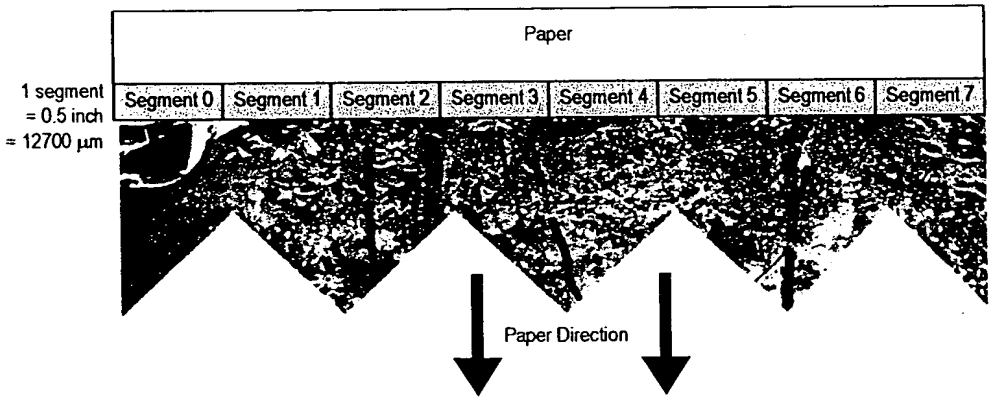


Figure 45. Arrangement of Segments in a 4-inch Printhead

An 8-inch printhead can be constructed from two 4-inch printheads or from a single 8-inch printhead consisting of 16 segments. Regardless of the construction mechanism, the effective printhead is still 8 inches in length.

A 2-inch printhead has a similar arrangement, but only uses 4 segments. Likewise, a full-bleed A4/Letter printer uses 17 segments for an effective 8.5 inch printing area.

Since the total number of nozzles in a segment is 800C (see Table 31), the total number of nozzles in a given printhead with S segments is 800CS. Thus segment N is responsible for printing dots 800N to 800N+799.

A number of considerations must be made when wiring up a printhead. As the width of the printhead increases, the number of segments increases, and the number of connections also increases. Each segment has its own ColorData connections (C of them), as well as SRClock and other connections for loading and printing.

9.2.1 Loading Considerations

When the number of segments *S* is small it is reasonable to load all the segments simultaneously by using a common SRClock line and placing C bits of data on each of the ColorData inputs for the segments. In a 4-inch printer, S=8, and therefore the total number of bits to transfer to the printhead in a single SRClock pulse is 32. However for an 8-inch printer, S=16, and it is unlikely to be reasonable to have 64 data lines running from the print data generator to the printhead.

Instead, it is convenient to group a number of segments together for loading purposes. Each group of segments is small enough to be loaded simultaneously, and share an SRClock. For example, an 8-inch printhead can have 2 segment groups, each segment group containing 8 segments. 32 ColorData lines can be shared for both groups, with 2 SRClock lines, one per segment group.

When the number of segment groups is not easily divisible, it is still convenient to group the segments. One example is a 8.5 inch printer for producing A4/Letter pages. There are 17 segments, and these can be grouped as two groups of 9 (9C bits of data going to each segment, with all 9C bits used in the first group, and only 8C bits used for the second group), or as 3 groups of 6 (again, C bits are unused in the last group).

As the number of segment groups increases, the time taken to load the printhead increases. When there is only one group, 800 load pulses are required (each pulse transfers C data bits). When there are G groups, 800G load pulses are required. The bandwidth of the connection between the data generator and the printhead must be able to cope and be within the allowable timing parameters for the particular application.

If *G* is the number of segment groups, and *L* is the largest number of segments in a group, the printhead requires LC ColorData lines and G SRClock lines. Regardless of G, only a single PTransfer line is required - it can be shared across all segments.

Since L segments in each segment group are loaded with a single SRClock pulse, any printing process must produce the data in the correct sequence for the printhead. As an example, when G=2 and L=4, the first SRClock1 pulse will transfer the ColorData bits for the next Print Cycle's dot 0, 800, 1600, and 2400. The first SRClock2 pulse will transfer the ColorData bits for the next Print Cycle's dot 3200, 4000, 4800, and 5600. The second SRClock1 pulse will transfer the ColorData bits for the next Print Cycle's dot 1, 801, 1601, and 2401. The second SRClock2 pulse will transfer the ColorData bits for the next Print Cycle's dot 3201, 4001, 4801 and 5601.

After 800G SRClock pulses (800 to each of SRClock1 and SRClock2), the entire line has been loaded into the printhead, and the common PTransfer pulse can be given.

It is important to note that the odd and even color outputs, although printed during the same Print Cycle, do not appear on the same physical output line. The physical separation of odd and even nozzles within the printhead, as well as separation between nozzles of different colors ensures that they will produce dots on different lines of the page. This relative difference must be accounted for when loading the data into the printhead. The actual difference in lines depends on the characteristics of the inkjet mechanism used in the printhead. The differences can be defined by variables D_1 and D_2 where D_1 is the distance between nozzles of different colors, and D_2 is the distance between nozzles of the same

color. Considering only a single segment group, Table 34 shows the dots transferred to segment *n* of a printhead during the first 4 pulses of the shared SRClock.

Table 34. Order of Dots Transferred to a Segment in a Printhead

Pulse	Dot	Color1 Line	Color2 Line	ColorC Line
1	800S[a]	N	N+D_1[b]	N+(C-1)D_1
2	800S+1	N+D_2[c]	N+D_1+D_2	N+(C-1)D_1+D_2
3	800S+2	N	N+D_1	N+(C-1)D_1
4	800S+3	N+D_2	N+D_1+D_2	N+(C-1)D_1+D_2

a. S = segment number

b. D_1 = number of lines between the nozzles of one color and the next (likely = 4-8)

c. D_2 = number of lines between two rows of nozzles of the same color (likely = 1)

And so on for all 800 SRClock pulses to the particular segment group.

9.2.2 Printing Considerations

With regards to printing, we print 4C nozzles from each segment in the low-speed printing mode, and 8C nozzles from each segment in the high speed printing mode.

While it is certainly possible to wire up segments in any way, this document only considers the situation where all segments fire simultaneously. This is because the low-speed printing mode allows low-power printing for small printheads (e.g. 2-inch and 4-inch), and the controller chip design assumes there is sufficient power available for the large print sizes (such as 8-18 inches). It is a simple matter to alter the connections in the printhead to allow grouping of firing should a particular application require it.

When all segments are fired at the same time 4CS nozzles are fired in the low-speed printing mode and 8CS nozzles are fired in the high-speed printing mode. Since all segments print simultaneously, the printing logic is the same as defined in Section 9.1.2.2 on page 67.

The timing for the two printing modes is therefore:

- 200 µs to print a line at low speed (comprised of 100 2µs cycles)
- 100 µs to print a line at high speed (comprised of 50 2µs cycles)

9.2.3 Feedback Considerations

A segment produces several lines of feedback, as defined in Section 9.1.3 on page 70. The feedback lines are used to adjust the timing of the firing pulses. Since multiple segments are collected together into a printhead, it is effective to share the feedback lines as a tri-state bus, with only one of the segments placing the feedback information on the feedback lines at a time.

Since the selection of which segment will place the feedback information on the shared Tsense, Vsense, Rsense, and Wsense lines uses the Color1Data line, the groupings of segments for loading data can be used for selecting the segment for feedback.

Just as there are G SRClock lines (a single line is shared between segments of the same segment group), there are G SenseSegSelect lines shared in the same way. When the correct SenseSegSelect line is pulsed, the segment of that group whose Color1Data bit is set will start to place data on the shared feedback lines. The segment previously active in terms of feedback must also be disabled by having a 0 on its Color1Data bit, and this segment may be in a different segment group. Therefore when there is more than one segment group. changing the feedback segment requires two steps: disabling the old segment, and enabling the new segment.

9.2.4 Printhead Connection Summary

This section assumes that a printhead has been constructed from a number of segments as described in the previous sections. It assumes that for data loading purposes, the segments have been grouped into G segment groups, with L segments in the largest segment group. It assumes there are C colors in the printhead. It assumes that the firing mechanism for the printhead is that all segments fire simultaneously, and only one segment at a time places feedback information on a common tri-state bus. Assuming all these things, Table 35 lists the external connections that are available from a printhead:

Table 35. Printhead Connections

Name	#Pins	Description
ChromapodSelect	3	Select which chromapod will fire (0-4)
NozzleSelect	4	Select which nozzle from the pod will fire (0-9)
PodgroupEnable	2	Enable the podgroups to fire (choice of: 01, 10, 11)
AEnable	1	Firing pulse for phasegroup A
BEnable	1	Firing pulse for phasegroup B
ColorData	CL	Inputs to C shift registers of segments 0 to L-1
SRClock	G	A pulse on SRClock[N] (ShiftRegisterClock N) loads the current values from ColorData lines into the L segments in segment group N.
PTransfer	1	Parallel transfer of data from the shift registers to the internal NozzleEnable bits (one per nozzle).
SenseSegSelect	G	A pulse on SenseSegSelect N ANDed with data on Color1Data[n] selects the sense lines for segment n in segment group N.
Tsense	1	Temperature sense
Vsense	1	Voltage sense
Rsense	1	Resistivity sense
Wsense	1	Width sense
Logic GND	1	Logic ground
Logic PWR	1	Logic power
V-	Bus bars	Actuator Ground
V+		Actuator Power
TOTAL	18+2G+CL	

10 Memjet Printhead Interface

The printhead interface (PHI) is the means by which the processor loads the Memjet printhead with the dots to be printed, and controls the actual dot printing process. The PHI contains:

- a LineSyncGen unit (LSGU), which provides synchronization signals for multiple chips (allows side-by-side printing and front/back printing) as well as stepper motors.
- a Memjet interface (MJI), which transfers data to the Memjet printhead, and controls the nozzle firing sequences during a print.
- a line loader/format unit (LLFU) which loads the dots for a given print line into local buffer storage and formats them into the order required for the Memjet printhead.

The units within the PHI are controlled by a number of registers that are programmed by the processor. In addition, the processor is responsible for setting up the appropriate parameters in the DMA controller for the transfers from memory to the LLFU. This includes loading white (all 0's) into appropriate colors during the start and end of a page so that the page has clean edges.

The PHI is capable of dealing with a variety of printhead lengths and formats. In terms of broad operating customizations, the PHI is parameterized as follows:

Table 36. Basic Printing Parameters

Name	Description	Range
MaxColors	No of Colors in printhead	1-4
SegmentsPerXfer	No of segments written to per transfer. Is equal to the number of segments in the largest segment group	1-9
SegmentGroups	No of segment groups in printhead	1-4

The internal structure of the PHI allows for a maximum of 4 colors, 9 segments per transfer, and 4 transfers. Transferring 4 colors to 9 segments is 36 bits per transfer, and 4 transfers to 9 segments equates to a maximum printed line length of 18 inches. The total number of dots per line printed by an 18-inch 4 color printhead is 115,200 (18 × 1600 × 4).

Other example settings are shown in Table 37:

Table 37. Example Settings for Basic Printing Parameters

PrinterLength	PrinterType	MaxColors	SegmentsPerXfer	SegmentGroups	BitsPerXfer	Comments
4-inch CMY	Photo	3	8	1	24	
8-inch CMYK	A4/Letter	4	8	2	32	
8.5 inch CMYK	A4/Letter full bleed	4	9	2	36	Last xfer not fully used
12 inch CMYK	A4 long / A3 short	4	8	3	32	
16 inch CMYK		4	8	4	32	
17 inch CMYK	A3 long full bleed	4	9	4	36	Last xfer not fully used
18 inch CMYK		4	9	4	36	

10.1 BLOCK DIAGRAM OF PRINTHEAD INTERFACE

The internal structure of the Printhead Interface is shown in Figure 46.

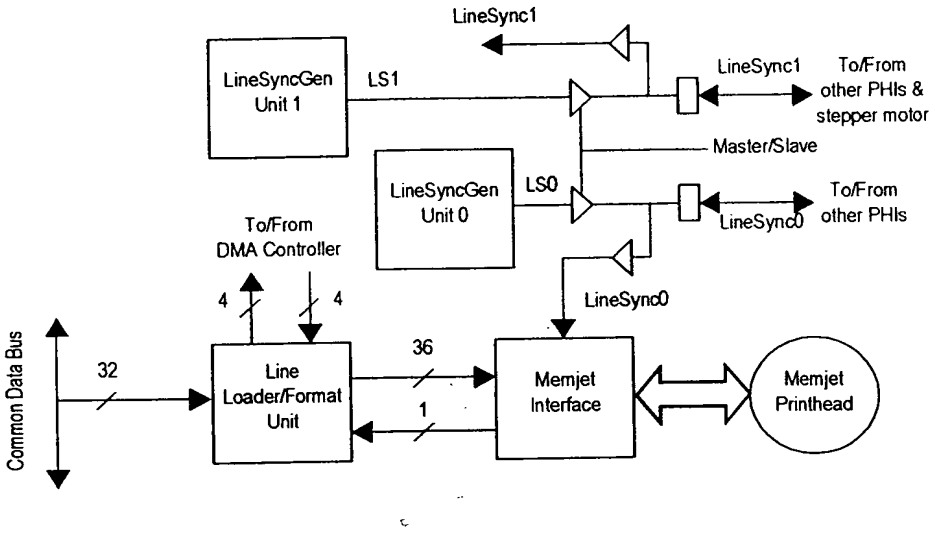


Figure 46. Internal Structure of Printhead Interface

In the PHI there are two LSGUs. The first LSGU produces LineSync0, which is used to control the Memjet Interface in all synchronized chips. The second LSGU produces LineSync1 which is used to pulse the paper drive stepper motor.

The Master/Slave pin on the chip allows multiple chips to be connected together for side-by-side printing, front/back printing etc. via a Master/Slave relationship. When the Master/Slave pin is attached to V_{DD}, the chip is considered to be the Master, and LineSync pulses generated by the two LineSyncGen units are enabled onto the two tri-state LineSync common lines (LineSync0 and LineSync1, shared by all the chips). When the Master/Slave pin is attached to GND, the chip is considered to be the Slave, and LineSync pulses generated by the two LineSyncGen units are not enabled onto the common LineSync lines. In this way, the Master chip's LineSync pulses are used by all PHIs on all the connected chips.

The following sections detail the LineSyncGen Unit, the Line Loader/Format Unit and Memjet Interface respectively.

10.2 LINESYNCGEN UNIT

The LineSyncGen units (LSGU) are responsible for generating the synchronization pulses required for printing a page. Each LSGU produces an external LineSync signal to enable line synchronization. The generator inside the LGSU generates a LineSync pulse when told to 'go', and then every so many cycles until told to stop. The LineSync pulse defines the start of the next line.

The exact number of cycles between LineSync pulses is determined by the CyclesBetweenPulses register, one per generator. It must be at least long enough to allow one line to print (100 µs or 200 µs depending on whether the speed is low or high) and another line to load, but can be longer as desired (for example, to accommodate special requirements of paper transport circuitry). If the CyclesBetweenPulses register is set to a number less than a line

print time, the page will not print properly since each LineSync pulse will arrive before the particular line has finished printing.

The following interface registers are contained in the LSGU:

Table 38. LineSyncGen Unit Registers

Register Name	Description
CyclesBetweenPulses	The number of cycles to wait between generating one LineSync pulse and the next.
Go	Controls whether the LSGU is currently generating LineSync pulses or not. A write of 1 to this register generates a LineSync pulse, transfers CyclesBetweenPulses to CyclesRemaining, and starts the countdown. When CyclesRemaining hits 0, another LineSync pulse is generated, CyclesBetweenPulses is transferred to CyclesRemaining and the countdown is started again. A write of 0 to this register stops the countdown and no more LineSync pulses are generated.
CyclesRemaining	A status register containing the number of cycles remaining until the next LineSync pulse is generated.

The LineSync pulse is not used directly from the LGSU. The LineSync pulse is enabled onto a tri-state LineSync line only if the Master/Slave pin is set to Master. Consequently the LineSync pulse is only used in the form as generated by the Master chip (pulses generated by Slave chips are ignored).

10.3 MEMJET INTERFACE

The Memjet interface (MJI) transfers data to the Memjet printhead, and controls the nozzle firing sequences during a print.

The MJI is simply a State Machine (see Figure 47) which follows the printhead loading and firing order described in Section 9.2.1 on page 73, Section 9.2.2 on page 74, and includes the functionality of the Preheat Cycle and Cleaning Cycle as described in Section 9.1.4 on page 70 and Section 9.1.5 on page 70. Both high-speed and low-speed printing modes are available, although the MJI always fires a given nozzle from all segments in a printhead simultaneously (there is no separate firing of nozzles from one segment and then others). Dot counts for each color are also kept by the MJI.

The MJI loads data into the printhead from a choice of 2 data sources:

- All 1s. This means that all nozzles will fire during a subsequent Print cycle, and is the standard mechanism for loading the printhead for a preheat or cleaning cycle.
- From the 36-bit input held in the Transfer register of the LLFU. This is the standard means of printing an image. The 36-bit value from the LLFU is directly sent to the printhead and a 1-bit 'Advance' control pulse is sent to the LLFU.

The MJI knows how many lines it has to print for the page. When the MJI is told to 'go', it waits for a LineSync pulse before it starts the first line. Once it has finished loading/printing a line, it waits until the next LineSync pulse before starting the next line. The MJI stops once the specified number of lines has been loaded/printed, and ignores any further LineSync pulses.

The MJI is therefore directly connected to the LLFU, LineSync0 (shared between all synchronized chips), and the external Memjet printhead. The basic structure is shown in Figure 47.

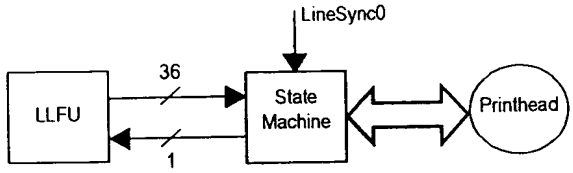


Figure 47. Memjet Interface

The MJI accepts 36 bits of data from the LLFU. Of these 36 bits, only the bits corresponding to the number of segments and number of colors will be valid. For example, if there are only 2 colors and 9 segments, bits 0-1 will be valid for segment 0, bits 2-3 will be invalid, bits 4-5 will be valid for segment 1, bits 6-7 will be invalid etc. The state machine does not care which bits are valid and which bits are not valid - it merely passes the bits out to the printhead. The data lines and control signals coming out of the MJI can be wired appropriately to the pinouts of the chip, using as few pins as required by the application range of the chip (see Section 10.3.1 on page 79 for more information).

10.3.1 Connections to Printhead

The MJI has a number of connections to the printhead, including a maximum of 4 colors, clocked in to a maximum of 9 segments per transfer to a maximum of 4 segment groups. The lines coming from the MJI can be directly connected to pins on the chip, although not all lines will always be pins. For example, if the chip is specifically designed for only connecting to 8 inch CMYK printers, only 32 bits of data need to be transferred each transfer pulse. Consequently 32 pins of data out (8 pins per color), and not 36 pins are required. In the same way, only 2 SRClock pulses are required, so only 2 pins instead of 4 pins are required to cater for the different SRClocks. And so on.

If the chip must be completely generic, then all connections from the MJI must be connected to pins on the chip (and thence to the Memjet printhead).

Table 39 lists the maximum connections from the MJI, many of which are always connected to pins on the chip. Where the number of pins is variable, a footnote explains what the number of pins depends upon. The sense of input and output is with respect to the MJI. The names correspond to the pin connections on the printhead.

Table 39. Memjet Interface Connections

Name	#Pins	I/O	Description
Chromapod Select	3	O	Select which chromapod will fire (0-4)
NozzleSelect	4	O	Select which nozzle from the pod will fire (0-9)
PodgroupEnable	2	O	Enable the podgroups to fire (choice of: 01, 10, 11)
AEnable	1	O	Firing pulse for podgroup A. In the current design all segments fire simultaneously, although multiple AEnable lines could be added for dividing the firing sequence over multiple segment groups for reasons of power and speed.

Table 39. Memjet Interface Connections

Name	#Pins	I/O	Description
BEnable	1	O	Firing pulse for podgroup B. In the current design all segments fire simultaneously, although multiple BEnable lines could be added for dividing the firing sequence over multiple segment groups for reasons of power and speed.
Color1Data[0-8]	9[a]	O	Output to Color1Data shift register of segments 0-8
Color2Data[0-8]	9[b]	O	Output to Color2Data shift register of segments 0-8
Color3Data[0-8]	9[c]	O	Output to Color3Data shift register of segments 0-8
Color4Data[0-8]	9[d]	O	Output to Color4Data shift register of segments 0-8
SRClock[1-4]	4[e]	O	A pulse on SRClock[N] (ShiftRegisterClock) loads the current values from Color1Data[0-8], Color2Data[0-8], Color3Data[0-8] and Color4Data[0-8] into the segment group N on the printhead.
PTransfer	1	O	Parallel transfer of data from the shift registers to the printhead's internal NozzleEnable bits (one per nozzle).
SenseSegSelect[1-4]	4[f]	O	A pulse on SenseSegSelect[N] ANDed with data on Color1Data[n] enables the sense lines for segment n in segment group N of the printhead.
Tsense	1	I	Temperature sense
Vsense	1	I	Voltage sense
Rsense	1	I	Resistivity sense
Wsense	1	I	Width sense
TOTAL	52		

a. Although 9 lines are available from the MJI, the number of pins coming from the chip will only reflect the actual number of segments in a segment group. The pins for Color1Data are mandatory, since each printhead must print in at *least* 1 color.

b. These lines are only translated into pins if the chip is to control a printhead with at least 2 colors. Although 9 lines are available from the MJI, the number of pins coming from the chip for Color2Data will only reflect the *actual* number of segments in a segment group.

c. These lines are only translated into pins if the chip is to control a printhead with at least 3 colors. Although 9 lines are available from the MJI, the number of pins coming from the chip for Color3Data will only reflect the *actual* number of segments in a segment group.

d. These lines are only translated into pins if the chip is to control a printhead with 4 colors. Although 9 lines are available from the MJI, the number of pins coming from the chip for Color4Data will only reflect the *actual* number of segments in a segment group.

e. Although 4 lines are available from the MJI, the number of pins coming from the chip will only reflect the *actual* number of segment groups. A minimum of 1 pin is required since there is at least 1 segment group (the entire printhead).

f. Although 4 lines are available from the MJI, the number of pins coming from the chip will only reflect the *actual* number of segment groups. A minimum of 1 pin is required since there is at least 1 segment group (the entire printhead).

10.3.2 Firing Pulse Duration

The duration of firing pulses on the AEnable and BEnable lines depend on the viscosity of the ink (which is dependant on temperature and ink characteristics) and the amount of power available to the printhead. The typical pulse duration range is 1.3 to 1.8 μs. The MJI therefore contains a programmable pulse duration table, indexed by feedback from the printhead. The table of pulse durations allows the use of a lower cost power supply, and aids in maintaining more accurate drop ejection.

The Pulse Duration table has 256 entries, and is indexed by the current Vsense and Tsense settings. The upper 4-bits of address come from Vsense, and the lower 4-bits of address

come from Tsense. Each entry is 8 bits, and represents a fixed point value in the range of 0-4µs. The process of generating the AEnable and BEnable lines is shown in Figure 48.

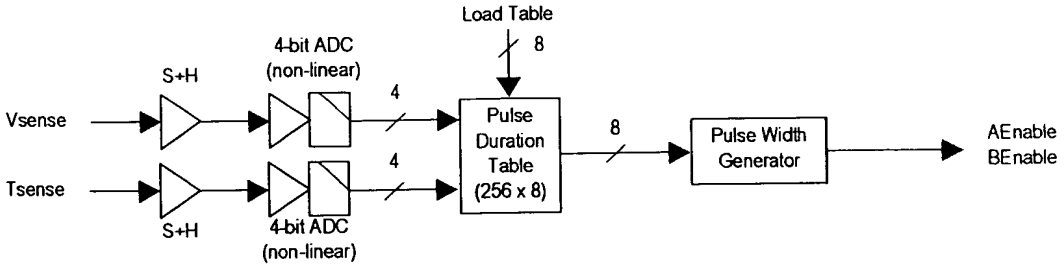


Figure 48. Generation of AEnable and BEnable Pulse Widths

The 256-byte table is written by the CPU before printing the first page. The table may be updated in between pages if desired. Each 8-bit pulse duration entry in the table combines:

- User brightness settings (from the page description)
- Viscosity curve of ink (from the QA Chip)
- Rsense
- Wsense
- Vsense
- Tsense

10.3.3 Dot Counts

The MJI maintains a count of the number of dots of each color fired from the printhead. The dot count for each color is a 32-bit value, individually cleared under processor control. At 32-bits length, each dot count can hold a maximum coverage dot count of 17 8-inch × 12-inch pages, although in typical usage, the dot count will be read and cleared after each page or half-page.

The dot counts are used by the processor to update the QA chip (see Section 7.5.4 on page 58) in order to predict when the ink cartridge runs out of ink. The processor knows the volume of ink in the cartridge for each of the colors from the QA chip. Counting the number of drops eliminates the need for ink sensors, and prevents the ink channels from running dry. An updated drop count is written to the QA chip after each page. A new page will not be printed unless there is enough ink left, and allows the user to change the ink without getting a dud half-printed page which must be reprinted.

The layout of the dot counter for Color1 is shown in Figure 49. The remaining 3 dot counters (Color1DotCount, Color2DotCount, and Color3DotCount) are identical in structure.

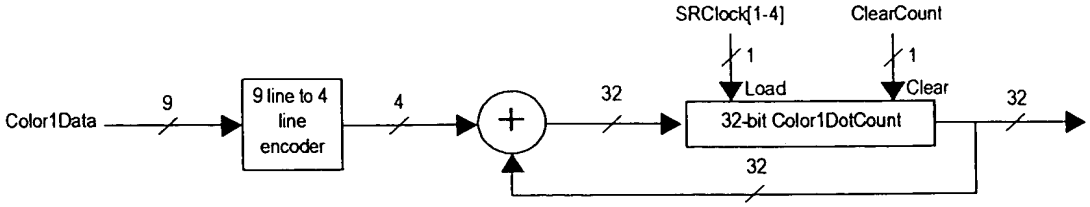


Figure 49. Dot Count Logic

10.3.4 Registers

The processor communicates with the MJI via a register set. The registers allow the processor to parameterize a print as well as receive feedback about print progress.

The following registers are contained in the MJI:

Table 40. Memjet Interface Registers

Register Name	Description
Print Parameters	
SegmentsPerXfer	The number of segments to write to each transfer. This also equals the number of cycles to wait between each transfer (before generating the next Advance pulse). Each transfer has MaxColors × SegmentsPerXfer valid bits.
SegmentGroups	The number of segment groups in the printhead. This equals the number of times that SegmentsPerXfer cycles must elapse before a single dot has been written to each segment of the printhead. The MJI does this 800 times to completely transfer all the data for the line to the printhead.
PrintSpeed	Whether to print at low or high speed (determines the value on the PodgroupEnable lines during the print).
NumLines	The number of Load/Print cycles to perform.
Monitoring the Print (read only from point of view of processor)	
Status	The Memjet Interface's Status Register
LinesRemaining	The number of lines remaining to be printed. Only valid while Go=1. Starting value is NumLines and counts down to 0.
TransfersRemaining	The number of sets of SegmentGroups transfers remaining before the Printhead is considered loaded for the current line. Starts at 800 and counts down to 0. Only valid while Go=1.
SegGroupsRemaining	The number of segment groups remaining in the current set of transfers of 1 dot to each segment. Starts at SegmentGroups and counts down to 0. Only valid while Go=1.
SenseSegment	The 9-bit value to place on the Color1Data lines during a subsequent feedback SenseSegSelect pulse. Only 1 of the 9 bits should be set, corresponding to one of the (maximum) 9 segments. See SenseSelect for how to determine which of the segment groups to sense.

Table 40. Memjet Interface Registers

Register Name	Description
SetAllNozzles	If non-zero, the 36-bit value written to the printhead during the LoadDots process is all 1s, so that all nozzles will be fired during the subsequent PrintDots process. This is used during the preheat and cleaning cycles. If 0, the 36-bit value written to the printhead comes from the LLFU. This is the case during the actual printing of regular images.
Actions	
Reset	A write to this register resets the MJI, stops any loading or printing processes, and loads all registers with 0.
SenseSelect	A write to this register with any value clears the FeedbackValid bit of the Status register, and the remaining action depends on the values in the LoadingDots and PrintingDots status bits. If either of the status bits are set, the Feedback bit is cleared and nothing more is done. If both status bits are clear, a pulse is given simultaneously on all 4 SenseSegSelect lines with all ColorData bits 0. This stops any existing feedback. Depending on the two low-order bits written to SenseSelect register, a pulse is given on SenseSegSelect1, SenseSegSelect2, SenseSegSelect3, or SenseSegSelect4 line, with the Color1Data bits set according to the SenseSegment register. Once the various sense lines have been tested, the values are placed in the Tsense, Vsense, Rsense, and Wsense registers, and the Feedback bit of the Status register is set.
Go	A write of 1 to this bit starts the LoadDots / PrintDots cycles, which commences with a wait for the first LineSync pulse. A total of NumLines lines are printed, each line being loaded/printed after the receipt of a LineSync pulse. The loading of each line consists of SegmentGroups 36-bit transfers. As each line is printed, LinesRemaining decrements, and TransfersRemaining is reloaded with SegmentGroups again. The status register contains print status information. Upon completion of NumLines, the loading/printing process stops, the Go bit is cleared, and any further LineSync pulses are ignored. During the final print cycle, nothing is loaded into the printhead. A write of 0 to this bit stops the print process, but does not clear any other registers.
ClearCounts	A write to this register clears the Color1DotCount, Color2DotCount, Color3DotCount, and Color4DotCount registers if bits 0, 1, 2, or 3 respectively are set. Consequently a write of 0 has no effect.
Feedback	
Tsense	Read only feedback of Tsense from the last SenseSegSelect pulse sent to segment SenseSegment. Is only valid if the FeedbackValid bit of the Status register is set.
Vsense	Read only feedback of Vsense from the last SenseSegSelect pulse sent to segment SenseSegment. Is only valid if the FeedbackValid bit of the Status register is set.
Rsense	Read only feedback of Rsense from the last SenseSegSelect pulse sent to segment SenseSegment. Is only valid if the FeedbackValid bit of the Status register is set.
Wsense	Read only feedback of Wsense from the last SenseSegSelect pulse sent to segment SenseSegment. Is only valid if the FeedbackValid bit of the Status register is set.
Color1DotCount	Read only 32-bit count of color1 dots sent to the printhead.
Color2DotCount	Read only 32-bit count of color2 dots sent to the printhead.

Table 40. Memjet Interface Registers

Register Name	Description
Color3DotCount	Read only 32-bit count of color3 dots sent to the printhead
Color4DotCount	Read only 32-bit count of color4 dots sent to the printhead

The MJI's Status Register is a 16-bit register with bit interpretations as follows:

Table 41. MJI Status Register

Name	Bits	Description
LoadingDots	1	If set, the MJI is currently loading dots, with the number of dots remaining to be transferred in TransfersRemaining. If clear, the MJI is not currently loading dots
PrintingDots	1	If set, the MJI is currently printing dots. If clear, the MJI is not currently printing dots.
PrintingA	1	This bit is set while there is a pulse on the AEnable line
PrintingB	1	This bit is set while there is a pulse on the BEnable line
FeedbackValid	1	This bit is set while the feedback values Tsense, Vsense, Rsense, and Wsense are valid.
Reserved	3	-
PrintingChromapod	4	This holds the current chromapod being fired while the PrintingDots status bit is set.
PrintingNozzles	4	This holds the current nozzle being fired while the PrintingDots status bit is set.

The following pseudocode illustrates the logic required to load a printhead for a single line. Note that loading commences only after the LineSync pulse arrives. This is to ensure the data for the line has been prepared by the LLFU and is valid for the first transfer to the printhead.

```
Wait for LineSync
For TransfersRemaining = 800 to 0
   For I = 0 to SegmentGroups
      If (SetAllNozzles)
         Set all ColorData lines to be 1
      Else
         Place 36 bit input on 36 ColorData lines
      EndIf
      Pulse SRClock[I]
      Wait SegmentsPerXfer cycles
      Send ADVANCE signal
   EndFor
EndFor
```

10.3.5 Preheat and Cleaning Cycles

The Cleaning and Preheat cycles are simply accomplished by setting appropriate registers in the MJI:

- SetAllNozzles = 1
- Set the PulseDuration register to either a low duration (in the case of the preheat mode) or to an appropriate drop ejection duration for cleaning mode.
- Set NumLines to be the number of times the nozzles should be fired
- Set the Go bit and then wait for the Go bit to be cleared when the print cycles have completed.

The LSGU must also be programmed to send LineSync pulses at the correct frequency.

10.4 LINE LOADER/FORMAT UNIT

The line loader/format unit (LLFU) loads the dots for a given print line into local buffer storage and formats them into the order required for the Memjet printhead. It is responsible for supplying the pre-calculated nozzleEnable bits to the Memjet interface for the eventual printing of the page.

The printing uses a double buffering scheme for preparing and accessing the dot-bit information. While one line is being loaded into the first buffer, the pre-loaded line in the second buffer is being read in Memjet dot order. Once the entire line has been transferred from the second buffer to the printhead via the Memjet interface, the reading and writing processes swap buffers. The first buffer is now read and the second buffer is loaded up with the new line of data. This is repeated throughout the printing process, as can be seen in the conceptual overview of Figure 50.

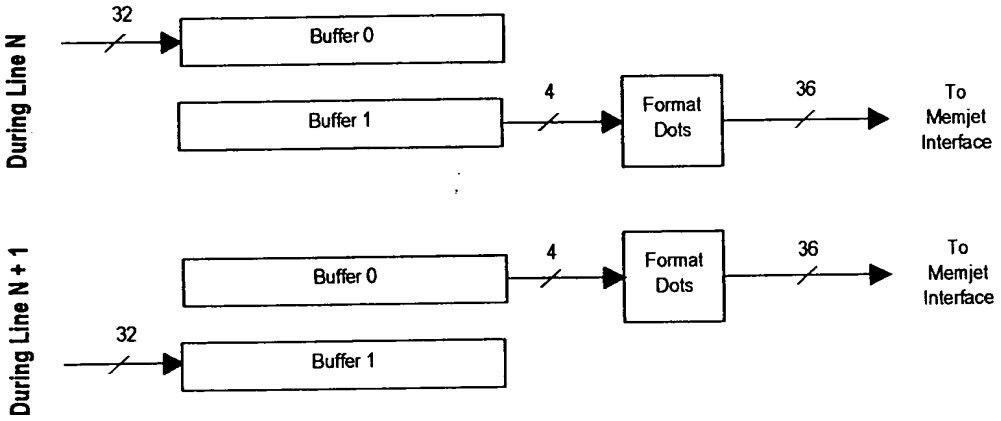


Figure 50. Conceptual Overview of Double Buffering During Print Lines N and N+1

The size of each buffer is 14KBytes to cater for the maximum line length of 18 inches in 4 colors (18 × 1600 × 4 bits = 115,200 bits = 14,400 bytes). The size for both Buffer 0 and Buffer 1 is 28.128 KBytes. While this design allows for a maximum print length of 18 inches, it is trivial to reduce the buffer size to target a specific application.

The actual implementation of the LLFU is shown in Figure 51. Since one buffer is being read from while the other is being written to, two sets of address lines must be used. The 32-bits DataIn from the common data bus are loaded depending on the WriteEnables, which are generated by the State Machine in response to the DMA Acknowledges.

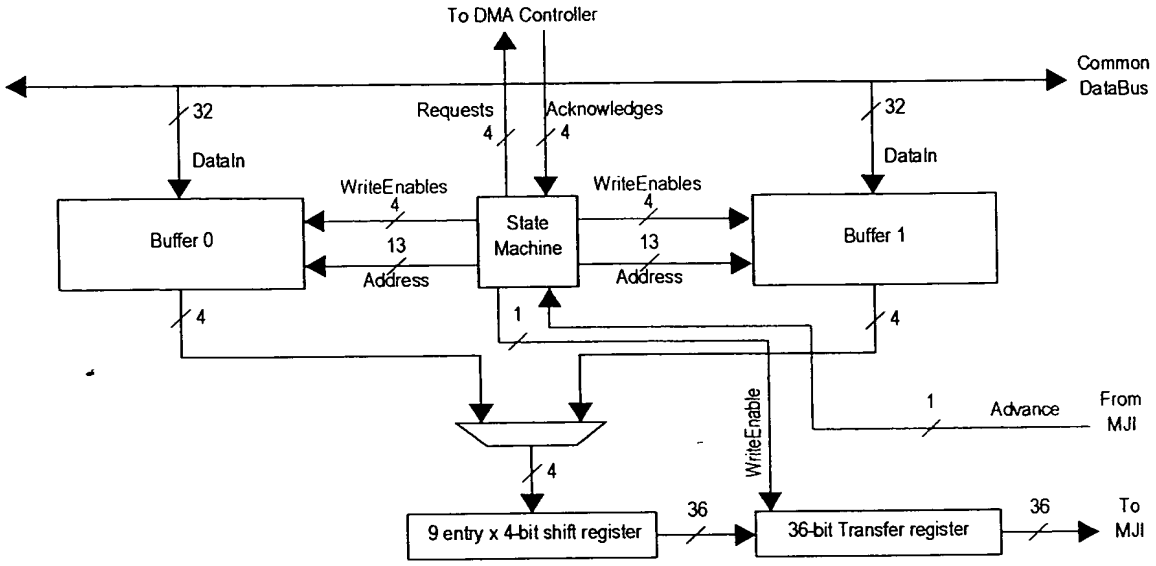


Figure 51. Structure of LLFU

A multiplexor chooses between the two 4-bit outputs of Buffer 0 and Buffer 1, and sends the result to a 9-entry by 4-bit shift register. After a maximum of 9 read cycles (the number depends on the number of segments written to per transfer), and whenever an Advance pulse comes from the MJI, the current 36-bit value from the shift register is gated into the 36-bit Transfer register, where it can be used by the MJI.

Note that not all the 36 bits are necessarily valid. The number of valid bits of 36 depends on the number of colors in the printhead, the number of segments, and the breakup of segment groups (if more than one segment group). For more information, see Section 9.2 on page 72.

A single line in an *L*-inch *C*-color printhead consists of ***1600L C***-color dots. At 1 bit per colored dot, a single print-line consists of ***1600LC bits***. The LLFU is capable of addressing a maximum line size of 18 inches in 4 colors, which equates to 108,800 bits (14 KBytes) per line. These bits must be supplied to the MJI in the correct order for being sent on to the printhead. See Section 9.2.1 on page 73 for more information concerning the Load Cycle dot loading order, but in summary, 2LC bits are transferred to the printhead in ***SegmentGroups*** transfers, with a maximum of 36 bits per transfer. Each transfer to a particular segment of the printhead must load all colors simultaneously.

10.4.1 Buffers

Each of the two buffers is broken into 4 sub-buffers, 1 per color. The size of each sub-buffer is 3600 bytes, enough to hold 18-inches of single color dots at 1600 dpi. The memory is accessed 32-bits at a time, so there are 900 addresses for each buffer (requiring 10 bits of address).

All the even dots are placed before the odd dots in each color's buffer, as shown in Figure 52. If there is any unused space it is placed at the end of each color's buffer.

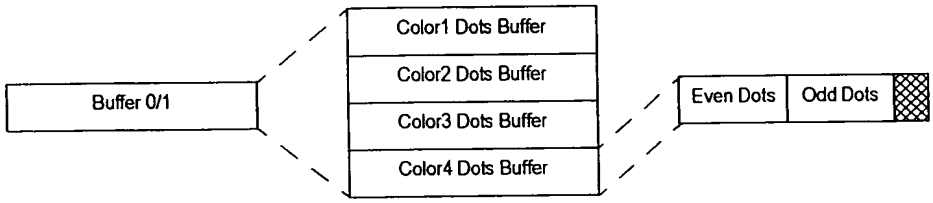


Figure 52. Conceptual Structure of Buffer

The *amount of memory* actually used is directly related to the printhead length. If the printhead is 18 inches, there are 1800 bytes of even dots followed by 1800 bytes of odd dots, with no unused space. If the printhead is 12 inches, there are 1200 bytes of even dots followed by 1200 odd dots, and 1200 bytes unused.

The *number of sub-buffers* gainfully used is directly related to the number of colors in the printhead. This number is typically 3 or 4, although it is quite feasible for this system to be used in a 1 or 2 color system (with some small memory wastage). In a desktop printing environment, the number of colors would be 4: Color1=Cyan, Color2=Magenta, Color3=Yellow, Color4=Black.

The addressing decoding circuitry is such that in a given cycle, a single 32-bit access can be made to all 4 sub-buffers - either a read from all 4 or a write to one of the 4. Only one bit of the 32-bits read from each color buffer is selected, for a total of 4 output bits. The process is shown in Figure 53. 15 bits of address allow the reading of a particular bit by means of 10-bits of address being used to select 32 bits, and 5-bits of address choose 1-bit from those 32. Since all color buffers share this logic, a single 15-bit address gives a total of 4 bits out, one per color. Each buffer has its own WriteEnable line, to allow a single 32-

bit value to be written to a particular color buffer in a given cycle. The 32-bits of DataIn are shared, since only one buffer will actually clock the data in.

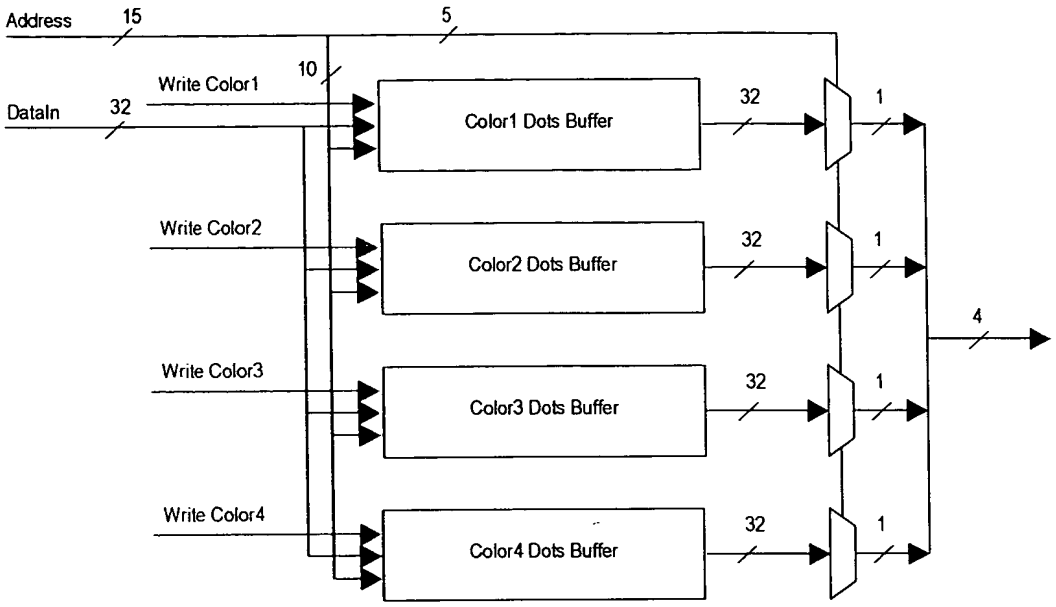


Figure 53. Logical Structure of Buffer

Note that regardless of the number of colors in the printhead, 4 bits are produced in a given read cycle (one bit from each color's buffer).

10.4.2 Address Generation

10.4.2.1 Reading

Address Generation for reading is straightforward. Each cycle we generate a bit address which is used to fetch 4 bits representing 1-bit per color for a particular segment. By adding 400 to the current bit address, we advance to the next segment's equivalent dot. We add 400 (not 800) since the odd and even dots are separated in the buffer. We do this firstly SegmentGroups sets of SegmentsPerXfer times to retrieve the data representing the even dots (the dot data is transferred to the MJI 36 bits at a time) and another SegmentGroups sets of SegmentsPerXfer times to load the odd dots. This entire process is repeated 400 times, incrementing the start address each time. Thus all dot values are transferred in the order required by the printhead in 400 × 2 × SegmentGroups × SegmentsPerXfer cycles.

In addition, we generate the TransferWriteEnable control signal. Since the LLFU starts before the MJI, we must transfer the first value before the Advance pulse from the MJI. We must also generate the next value in readiness for the first Advance pulse. The solution is to transfer the first value to the Transfer register after SegmentsPerXfer cycles, and then to stall SegmentsPerXfer-cycles later, waiting for the Advance pulse to start the next SegmentsPerXfer cycle group. Once the first Advance pulse arrives, the LLFU is synchronized to the MJI. However, the LineSync pulse to start the next line must arrive at the MJI at least 2SegmentsPerXfer cycles after the LLFU so that the initial Transfer value is valid and the next 32-bit value is ready to be loaded into the Transfer register.

The read process is shown in the following pseudocode:

```
DoneFirst = FALSE
For DotInSegment0 = 0 to 400
   CurrAdr = DotInSegment0
   XfersRemaining = 2 x SegmentGroups
   DotCount = SegmentsPerXfer
   Do
      V1 = DotCount = 0
      TransferWriteEnable = (V1 AND NOT DoneFirst) OR ADVANCE
      Stall = V1 AND (NOT TransferWriteEnable)
      If (NOT Stall)
         Shift Register=Fetch 4-bits from CurrReadBuffer:CurrAdr
         CurrAdr = CurrAdr + 400
         If (V1)
            DotCount = SegmentsPerXfer - 1
            XfersRemaining = XfersRemaining - 1
         Else
            DotCount = DotCount - 1
         EndIf
      EndIf
   Until (XfersRemaining=0) AND (NOT Stall)
EndFor
```

The final transfer may not be fully utilized. This occurs when the number of segments per transfer does not divide evenly into the actual number of segments in the printhead. An example of this is the 8.5 inch printhead, which has 17 segments. Transferring 9 segments each time means that only 8 of the last 9 segments will be valid. Nonetheless, the timing requires the entire 9th segment value to be generated (even though it is not used). The actual address is therefore a don't care state since the data is not used.

Once the line has finished, the CurrReadBuffer value must be toggled by the processor.

10.4.2.2 Writing

The write process is also straightforward. 4 DMA request lines are output to the DMA controller. As requests are satisfied by the return DMA Acknowledge lines, the appropriate 8-bit destination address is selected (the lower 5 bits of the 15-bit output address are *don't care* values) and the acknowledge signal is passed to the correct buffer's WriteEnable control line (the Current Write Buffer is ¬CurrentReadBuffer). The 10-bit destination address is selected from the 4 current addresses, one address per color. As DMA requests are satisfied the appropriate destination address is incremented, and the corresponding TransfersRemaining counter is decremented. The DMA request line is only set when the number of transfers remaining for that color is non-zero.

The following pseudocode illustrates the Write process:

```
CurrentAdr[1-4] = 0
While (ColorXfersRemaining[1-4] are non-zero)
   DMARequest[1-4] = ColorXfersRemaining[1-4] NOT = 0
   If DMAAknowledge[N]
      CurrWriteBuffer:CurrentAdr[N] = Fetch 32-bits from data bus
      CurrentAdr[N] = CurrentAdr[N] + 1
      ColorXfersRemaining[N] = ColorXfersRemaining[N] - 1 (floor 0)
   EndIf
EndWhile
```

10.4.3 Registers

The following interface registers are contained in the LLFU:

Table 42. Line Load/Format Unit Registers

Register Name	Description
SegmentsPerXfer	The number of segments whose dots must be loaded before each transfer. This has a maximum value of 9.
SegmentGroups	The number of segment groups in the printhead. This has a maximum number of 4.
CurrentReadBuffer	The current buffer being read from. When Buffer0 is being read from, Buffer1 is written to and vice versa. Should be toggled with each AdvanceLine pulse from the MJI.
Go	Bits 0 and 1 control the starting of the read and write processes respectively. A non-zero write to the appropriate bit starts the process.
Stop	Bits 0 and 1 control the stopping of the read and write processes respectively. A non-zero write to the appropriate bit stops the process.
Stall	This read-only status bit comes from the LLFU's Stall flag. The Stall bit is valid when the write Go bit is set. A Stall value of 1 means that the LLFU is waiting for the ADVANCE pulse from the MJI to continue. The CPU can safely start the LSGU for the first line once the Stall bit is set.
ColorXfersRemaining[1-4]	The number of 32-bit transfers remaining to be read into the specific Color[N] buffer.

10.5 CONTROLLING A PRINT

When controlling a print the CPU programs and starts the LLFU in read mode to ensure that the first line of the page is transferred to the buffer. When the interrupts arrive from the DMA controller, the CPU can switch LLFU buffers, and program the MJI. The CPU then starts the LLFU in read/write mode and starts the MJI. The CPU should then wait a sufficient period of time to ensure that other connected printer controllers have also started their LLFUs and MJIs (if there are no other connected printer controllers, the CPU must wait until the Stall bit of the LLFU is set, a duration of 2SegmentsPerXfer cycles). The CPU can then program the LGSU to start the synchronized print. As interrupts arrive from the DMA controllers, the CPU can reprogram the DMA channels, swap LLFU buffers, and restart the LLFU in read/write mode. Once the LLFU has effectively filled its pipeline, it will stall until the next Advance pulse from the MJI. The MJI does not have to be touched during the print.

If for some reason the CPU wants to make any changes to the MJI or LLFU registers during an inter-line period it should ensure that the current line has finished printing/loading by polling the status bits of the MJI and the Go bits of the LLFU.

11 Generic Printer Driver

This section describes generic aspects of any host-based printer driver for CePrint.

11.1 Graphics and Imaging Model

We assume that the printer driver is closely coupled with the host graphics system, so that the printer driver can provide device-specific handling for different graphics and imaging operations, in particular compositing operations and text operations.

We assume that the host provides support for color management, so that device-independent color can be converted to CePrint-specific CMYK color in a standard way, based on a user-selected CePrint-specific ICC (International Color Consortium) color profile. The color profile is normally selected implicitly by the user when the user specifies the output medium in the printer (i.e. plain paper, coated paper, transparency, etc.). The page description sent to the printer always contains *device-specific* CMYK color.

We assume that the host graphics system renders images and graphics to a nominal resolution specified by the printer driver, but that it allows the printer driver to take control of rendering text. In particular, the graphics system provides sufficient information to the printer driver to allow it to *render and position* text at a higher resolution than the nominal device resolution.

We assume that the host graphics system requires random access to a contone page buffer at the nominal device resolution, into which it composites graphics and imaging objects, but that it allows the printer driver to take control of the actual compositing - i.e. it expects the printer driver to *manage* the page buffer.

11.2 Two-Layer Page Buffer

The printer's page description contains a 267 ppi contone layer and an 800 dpi black layer. The black layer is conceptually *above* the contone layer, i.e. the black layer is composited *over* the contone layer by the printer. The printer driver therefore maintains a page buffer which correspondingly contains a low-resolution contone layer and a high-resolution black layer.

The graphics systems renders and composites objects into the page buffer bottom-up - i.e. later objects obscure earlier objects. This works naturally when there is only a single layer, but not when there are two layers which will be composited later. It is therefore necessary to detect when an object being placed on the contone layer obscures something on the black layer.

When obscuration is detected, the obscured black pixels are composited with the contone layer and removed from the black layer. The obscuring object is then laid down on the contone layer, possibly interacting with the black pixels in some way. If the compositing mode of the obscuring object is such that no interaction with the background is possible, then the black pixels can simply be discarded without being composited with the contone layer. In practice, of course, there is little interaction between the contone layer and the black layer.

The printer driver specifies a nominal page resolution of 267 ppi to the graphics system. Where possible the printer driver relies on the graphics system to render image and graphics objects to the pixel level at 267 ppi, with the exception of *black* text. The printer driver

fields all text rendering requests, detects and renders black text at 800 dpi, but returns non-black text rendering requests to the graphics system for rendering at 267 ppi.

Ideally the graphics system and the printer driver manipulate color in device-independent RGB, deferring conversion to device-specific CMYK until the page is complete and ready to be sent to the printer. This reduces page buffer requirements and makes compositing more rational. Compositing in CMYK color space is not ideal.

Ultimately the graphics system asks the printer driver to composite each rendered object into the printer driver's page buffer. Each such object uses 24-bit contone RGB, and has an explicit (or implicitly opaque) opacity channel.

The printer driver maintains the two-layer page buffer in three parts. The first part is the low-resolution (267 ppi) contone layer. This consists of a 24-bit RGB bitmap. The second part is a low-resolution black layer. This consists of an 8-bit opacity bitmap. The third part is a high-resolution (800 dpi) black layer. This consists of a 1-bit opacity bitmap. The low-resolution black layer is a subsampled version of the high-resolution opacity layer. In practice, assuming the low resolution is an integer factor n of the high resolution (e.g. n = 800 / 267 = 3), each low-resolution opacity value is obtained by averaging the corresponding $n \times n$ high-resolution opacity values. This corresponds to box-filtered subsampling. The subsampling of the black pixels effectively antialiases edges in the high-resolution black layer, thereby reducing ringing artifacts when the contone layer is subsequently JPEG-compressed and decompressed.

The structure and size of the page buffer is illustrated in Figure 54.

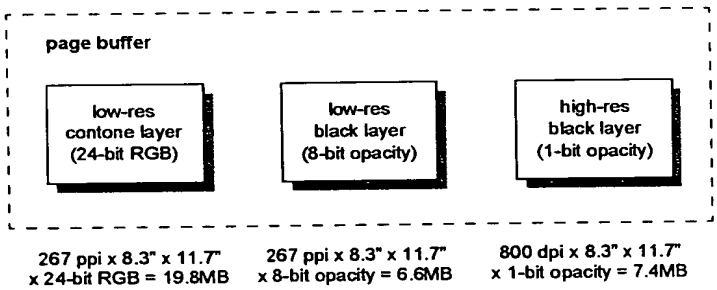


Figure 54. Two-layer page buffer

11.3 COMPOSITING MODEL

For the purposes of discussing the page buffer compositing model, we define the following variables.

Table 43. Compositing variables

variable	description	resolution	format
n	low to high resolution scale factor	-	-
C_C	contone layer color	low	8-bit color component
C_G	contone object color	low	8-bit color component
α_G	contone object opacity	low	8-bit opacity
α_L	low-resolution black layer opacity	low	8-bit opacity

Table 43. Compositing variables

variable	description	resolution	format
α_B	black layer opacity	high	1-bit opacity
α_T	black object opacity	high	1-bit opacity

When a black object of opacity α_T is composited with the black layer, the black layer is updated as follows:

$$\alpha_{B_{x,y}} = \alpha_{B_{x,y}} \vee \alpha_{T_{x,y}} \quad (1)$$

$$\alpha_{L_{x,y}} = \frac{1}{n^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} 255\alpha_{B_{nx+i,ny+j}} \quad (2)$$

The object opacity is simply *or*ed with the black layer opacity (Eq. 1), and the corresponding part of the low-resolution black layer is re-computed from the high-resolution black layer (Eq. 2).

When a contone object of color C_G and opacity α_G is composited with the contone layer, the contone layer and the black layer are updated as follows:

$$C_{C_{x,y}} = C_{C_{x,y}}(1-\alpha_{L_{x,y}}) \text{ if } (\alpha_{G_{x,y}} > 0) \quad (3)$$

$$\alpha_{L_{x,y}} = 0 \text{ if } (\alpha_{G_{x,y}} > 0) \quad (4)$$

$$\alpha_{B_{x,y}} = 0 \text{ if } (\alpha_{G_{x/n,y/n}} > 0) \quad (5)$$

$$C_{C_{x,y}} = C_{C_{x,y}}(1-\alpha_{G_{x,y}}) + C_{G_{x,y}}\alpha_{G_{x,y}} \quad (6)$$

Wherever the contone object hides the black layer, even if not fully opaquely, the affected black layer pixels are composited with the contone layer (Eq. 3) and removed from the black layer (Eq. 4 and Eq. 5). The contone object is then composited with the contone layer (Eq. 6).

11.4 PAGE COMPRESSION AND DELIVERY

Once page rendering is complete, the printer driver converts the contone layer to CePrint-specific CMYK with the help of color management functions provided by the graphics system.

The printer driver then compresses and packages the black layer and the contone layer into a CePrint page description as described in Section 6.2. This page description is delivered to the printer via the standard spooler.

Note that the black layer is manipulated as a set of 1-bit opacity values, but is delivered to the printer as a set of 1-bit black values. Although these two interpretations are different, they share the same representation, and so no data conversion is required.

The forward discrete cosine transform (DCT) is the costliest part of JPEG compression. In current high-quality software implementations, the forward DCT of each 8×8 block

requires 12 integer multiplications and 32 integer additions [6]. On typical modern general-purpose processors, an integer multiplication requires 10 cycles, and an integer addition requires 2 cycles [3,11]. This equates to a total cost per block of 184 cycles.

The 26.4MB contone layer consists of 432,538 JPEG blocks, giving an overall forward DCT cost of about 80Mcycles. At 150MHz this equates to about 0.5 seconds, which is 25% of the 2 second rendering time allowed per page.

A CE-oriented processor may have DSP support, in which case the presence of single-cycle multiplication makes the JPEG compression time negligible.

11.5 BANDED OUTPUT

The printer control protocol supports the transmission of the page to the printer as a series of bands. If the graphics system also supports banded output, then this allows the printer driver to reduce its memory requirements by rendering the image one band at a time. Note, however, that rendering one band at a time can be more expensive than rendering the whole page at once, since objects which span multiple bands have to be handled multiple times.

Although banded rendering can be used to reduce memory requirements in the printer driver, buffers for two bands are still required. One buffer is required for the band being transmitted to the printer; another buffer is required for the band being rendered. A single buffer may suffice if the connection between the host processor and printer is sufficiently fast. The band being transmitted to the printer may also be stored on disk, if a disk drive is present in the system, and only loaded into memory block-by-block during transmission.

12 Windows 9x/NT/CE Printer Driver

12.1 Windows 9x/NT/CE Printing System

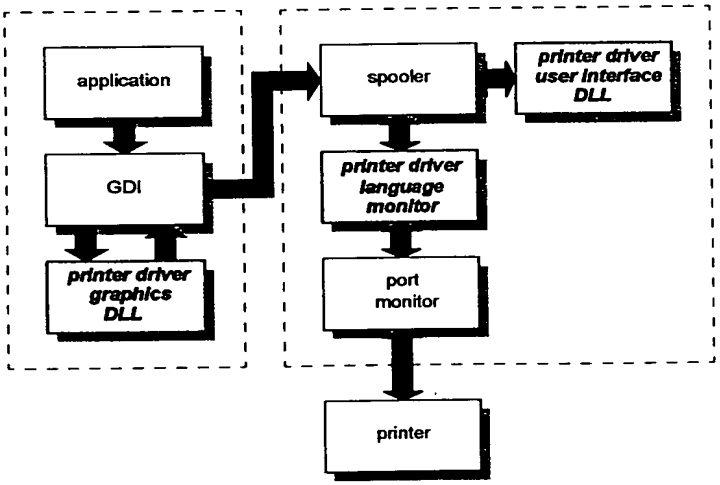
In the Windows 9x/NT/CE printing system [7,8], a printer is a *graphics device*, and an application communicates with it via the *graphics device interface* (GDI). The printer driver *graphics DLL* (dynamic link library) implements the device-dependent aspects of the various graphics functions provided by GDI.

The *spooler* handles the delivery of pages to the printer, and may reside on a different machine to the application requesting printing. It delivers pages to the printer via a *port monitor* which handles the physical connection to the printer. The optional *language monitor* is the part of the printer driver which imposes additional protocol on communication with the printer, and in particular decodes status responses from the printer on behalf of the spooler.

The printer driver *user interface DLL* implements the user interface for editing printer-specific properties and reporting printer-specific events.

The structure of the Windows 9x/NT/CE printing system is illustrated in Figure 55.

Figure 55. Windows 9x/NT/CE printing system with printer driver components indicated



The printer driver language monitor and user interface DLL must implement the implement the relevant aspects of the printer control protocol described in Section 6.

The remainder of this section describes the design of the printer driver graphics DLL. It should be read in conjunction with the appropriate Windows 9x/NT/CE DDK documentation [7,8].

12.2 WINDOWS 9x/NT/CE GRAPHICS DEVICE INTERFACE (GDI)

GDI provides functions which allow an application to draw on a *device surface*, i.e. typically an abstraction of a display screen or a printed page. For a raster device, the device surface is conceptually a color bitmap. The application can draw on the surface in a device-independent way, i.e. independently of the resolution and color characteristics of the device.

The application has random access to the entire device surface. This means that if a memory-limited printer device requires banded output, then GDI must buffer the entire page's GDI commands and replay them windowed into each band in turn. Although this provides the application with great flexibility, it can adversely affect performance.

GDI supports color management, whereby device-independent colors provided by the application are transparently translated into device-dependent colors according to a standard ICC (International Color Consortium) color profile of the device. A printer driver can activate a different color profile depending, for example, on the user's selection of paper type on the driver-managed printer property sheet.

GDI supports line and spline outline graphics (paths), images, and text. Outline graphics, including outline font glyphs, can be stroked and filled with bit-mapped brush patterns. Graphics and images can be geometrically transformed and composited with the contents of the device surface. While Windows 95/NT4 provides only boolean compositing operators, Windows 98/NT5 provides proper alpha-blending [8].

12.3 PRINTER DRIVER GRAPHICS DLL

A raster printer can, in theory, utilize standard printer driver components under Windows 9x/NT/CE, and this can make the job of developing a printer driver trivial. This relies on being able to model the device surface as a single bitmap. The problem with this is that text and images must be rendered at the same resolution. This either compromises text resolution, or generates too much output data, compromising performance.

As described earlier, CePrint's approach is to render black text and images at different resolutions, to optimize the reproduction of each. The printer driver is therefore implemented according to the generic design described in Section 11.

The driver therefore maintains a two-layer three-part page buffer as described in Section 11.2, and this means that the printer driver must take over managing the device surface, which in turn means that it must mediate all GDI access to the device surface.

12.3.1 Managing the Device Surface

A graphics driver must support a number of standard functions, including the following:

Table 44. Standard graphics driver interface functions

function	description
`DrvEnableDriver`	Initial entry point into the driver graphics DLL. Returns addresses of functions supported by the driver.
`DrvEnablePDEV`	Creates a logical representation of a physical device with which the driver can associate a drawing surface.
`DrvEnableSurface`	Creates a surface to be drawn on, associated with a given PDEV.

`DrvEnablePDEV` indicates to GDI, via the `flGraphicsCaps` member of the returned `DEVINFO` structure, the graphics rendering capabilities of the driver. This is discussed further below.

`DrvEnableSurface` creates a device surface consisting of two conceptual layers and three parts: the 267 ppi contone layer 24-bit RGB color, the 267 ppi black layer 8-bit opacity, and the 800 dpi black layer 1-bit opacity. The *virtual* device surface which encapsulates these two layers has a nominal resolution of 267 ppi, so this is the resolution at which GDI operations take place.

Although the aggregate page buffer requires about 34MB of memory, the size of the page buffer can be reduced arbitrarily by rendering the page one band at a time, as described in Section 12.3.4.

A printer-specific graphics driver must also support the following functions:

Table 45. Required printer driver functions

function	description
`DrvStartDoc`	Performs any start-of-document handling
`DrvStartPage`	Handles the start of a new page.
`DrvSendPage`	Sends the current page to the printer via the spooler.
`DrvEndDoc`	Performs any end-of-document handling.

`DrvStartDoc` sends the *start document* command to the printer, and `DrvEndDoc` sends the *end document* command.

`DrvStartPage` sends the *start page* command with the page header to the printer.

`DrvSendPage` converts the contone layer from RGB to CMYK using GDI-provided color management functions, compresses both the contone and black layers, and sends the compressed page as a single band to the printer (in a *page band* command).

Both `DrvStartPage` and `DrvSendPage` use `EngWritePrinter` to send data to the printer via the spooler.

Managing the device surface and mediating GDI access to it means that the printer driver must support the following additional functions:

Table 46. Required graphics driver functions for a device-managed surface

function	description
`DrvCopyBits`	Translates between device-managed raster surfaces and GDI-managed standard-format bitmaps.
`DrvStrokePath`	Srokes a path.
`DrvPaint`	Paints a specified region.
`DrvTextOut`	Renders a set of glyphs at specified positions.

Copying images, stroking paths and filling regions all occur on the contone layer, while rendering solid black text occurs on the bi-level black layer. Furthermore, rendering non-black text also occurs on the contone layer, since it isn't supported on the black layer.

Conversely, stroking or filling with solid black can occur on the black layer (if we so choose).

Although the printer driver is obliged to *hook* the aforementioned functions, it can *punt* function calls which apply to the contone layer back to the corresponding GDI implementations of the functions, since the contone layer is a standard-format bitmap. For every `DrvXxx` function there is a corresponding `EngXxx` function provided by GDI.

As described in Section 11.2, when an object destined for the contone layer obscures pixels on the black layer, the obscured black pixels must be transferred from the black layer to the contone layer before the contone object is composited with the contone layer. The key to this process working is that obscuration is detected and handled in the hooked call, *before* it is punted back to GDI. This involves determining the pixel-by-pixel opacity of the contone object from its geometry, and using this opacity to selectively transfer black pixels from the black layer to the contone layer as described in Section 11.2.

12.3.2 Determining Contone Object Geometry

It is possible to determine the geometry of each contone object before it is rendered and thus determine efficiently which black pixels it obscures. In the case of `DrvCopyBits` and `DrvPaint`, the geometry is determined by a clip object (`CLIPOBJ`), which can be enumerated as a set of rectangles.

In the case of `DrvStrokePath`, things are more complicated. `DrvStrokePath` supports both straight-line and Bézier-spline curve segments, and single-pixel-wide lines and geometric-wide lines. The first step is to avoid the complexity of Bézier-spline curve segments and geometric-wide lines altogether by clearing the corresponding capability flags (`GCAPS_BEZIERS` and `GCAPS_GEOMETRICWIDE`) in the `flGraphicsCaps` member of the driver's `DEVINFO` structure. This causes GDI to reformulate such calls as sets of simpler calls to `DrvPaint`. In general, GDI gives a driver the opportunity to accelerate high-level capabilities, but *simulates* any capabilities not provided by the driver.

What remains is simply to determine the geometry of a single-pixel-wide straight line. Such a line can be solid or *cosmetic*. In the latter case, the line style is determined by a styling array in the specified line attributes (`LINEATTRS`). The styling array specifies how the line alternates between being opaque and transparent along its length, and so supports various dashed line effects etc.

When the brush is solid black, straight lines can also usefully be rendered to the black layer, though with the increased width implied by the 800 dpi resolution.

12.3.3 Rendering Text

In the case of a `DrvTextOut`, things are also more complicated. Firstly, the opaque background, if any, is handled like any other fill on the contone layer (see `DrvPaint`). If the foreground brush is not black, or the mix mode is not effectively opaque, or the font is not scalable, or the font indicates outline stroking, then the call is punted to `EngTextOut`, to be applied to the contone layer. Before the call is punted, however, the driver determines the geometry of each glyph by obtaining its bitmap (via `FONTOBJ_cGetGlyphs`), and makes the usual obscuration check against the black layer.

If punting a `DrvTextOut` call is not allowed (the documentation is ambiguous), then the driver should disallow complex text operations. This includes disallowing outline stroking

(by clearing the GCAPS_VECTOR_FONT capability flag), and disallowing complex mix modes (by clearing the GCAPS_ARBMIXTXT capability flag).

If the foreground brush is black and opaque, and the font is scalable and not stroked, then the glyphs are rendered on the black layer. In this case the driver determines the geometry of each glyph by obtaining its *outline* (again via FONTOBJ_cGetGlyphs, but as a PATHOBJ). The driver then renders each glyph from its outline at 800 dpi and writes it to the black layer. Although the outline geometry uses device coordinates (i.e. at 267 ppi), the coordinates are in fixed point format with plenty of fractional precision for higher-resolution rendering.

Note that strikethrough and underline rectangles are added to the glyph geometry, if specified.

The driver must set the GCAPS_HIGHRESTEXT flag in the DEVINFO to request that glyph positions (again in 267 ppi device coordinates) be supplied by GDI in high-precision fixed-point format, to allow accurate positioning at 800 dpi. The driver must also provide an implementation of the DrvGetGlyphMode function, so that it can indicate to GDI that glyphs should be cached as outlines rather than bitmaps. Ideally the driver should cache rendered glyph bitmaps for efficiency, memory allowing. Only glyphs below a certain point size should be cached.

12.3.4 Banded Output

As described in Section 6, the printer control protocol supports banded output by breaking the page description into a page header and a number of page bands. GDI supports banded output to a printer to cater for printer drivers and printers which have limited internal buffer memory.

GDI can handle banded output without application involvement. GDI simply records all the graphics operations performed by the application in a metafile, and then replays the entire metafile to the printer driver for each band in the page. The printer driver must clip the graphics operations to the current band, as usual. Banded output can be more efficient if the application takes note of the RC_BANDING bit in the driver's raster capabilities (returned by GetDeviceCaps when called with the RASTERCAPS index) and only performs graphics operations relevant to each band.

If banded output is desired because memory is limited, then the printer driver must enable banding by calling EngMarkBandingSurface in DrvEnableSurface. It must also support the following additional functions:

Table 47. Required printer driver functions

function	description
DrvStartBanding	Prepares the driver for banding and returns the origin of the first band.
DrvNextBand	Sends the current band to the printer and returns the origin of the next band, if any.

Like DrvSendPage, DrvNextBand converts the contone layer from RGB to CMYK using GDI-provided color management functions, compresses both the contone and black layers, and sends the compressed page band to the printer (in a *page band* command).

It uses EngWritePrinter to send the band data to the printer via the spooler.

13 References

[1] ANSI/EIA 538-1988, *Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Equipment*, August 1988

[2] Farrell, J., "How to Allocate Bits to Optimize Photographic Image Quality", Proceedings of IS&T International Conference on Digital Printing Technologies, 1998, pp.572-576

[3] Heinrich, J., *MIPS R4000 Microprocessor User's Manual*, PTR Prentice Hall, 1993

[4] Humphreys, G.W., and V. Bruce, *Visual Cognition*, Lawrence Erlbaum Associates, 1989, p.15

[5] ISO/IEC 19018-1:1994, *Information technology - Digital compression and coding of continuous-tone still images: Requirements and guidelines*, 1994

[6] Loeffler, C., A. Ligtenberg and G. Moschytz, "Practical Fast 1-D DCT Algorithms with 11 Multiplications", Proceedings of the International Conference on Acoustics, Speech, and Signal Processing 1989 (ICASSP '89), pp.988-991

[7] Microsoft Corp., *Microsoft Windows NT 4.0 Device Driver Kit*, 1997

[8] Microsoft Corp., *Microsoft Windows NT 5.0 Device Driver Kit*, 1998

[9] Murray, J.D., and van Ryper, W., *Encyclopedia of Graphics File Formats*, O'Reilly & Associates, First Ed., 1994

[10] Olsen, J. "Smoothing Enlarged Monochrome Images", in Glassner, A.S. (ed.), *Graphics Gems*, AP Professional, 1990

[11] Schmit, M.L., *Pentium Processor Optimization Tools*, AP Professional, 1995

[12] Silverbrook Research, *Authentication Chip*, 1998

[13] Silverbrook Research, *Authentication of Consumables*, 1998

[14] Silverbrook Research, *Memjet*, 1998

[15] Thompson, H.S., *Multilingual Corpus 1* CD-ROM, European Corpus Initiative

[16] Urban, S.J., "Review of standards for electronic imaging for facsimile systems", Journal of Electronic Imaging, Vol.1(1), January 1992, pp.5-21

[17] Wallace, G.K., "The JPEG Still Picture Compression Standard", *Communications of the ACM*, 34(4), April 1991, pp.30-44

[18] Yasuda, Y., "Overview of Digital Facsimile Coding Techniques in Japan", Proceedings of the IEEE, Vol. 68(7), July 1980, pp.830-845